Tuesday, December 24, 2024 12:00 AM

Home - JOBS GATEWAY
JOBS GATEWAY

Overview - Workday

1964_WEB_myScheduling_Production_Production_ESOPortal
1964_WEB_myScheduling_Production_Production_ESOPortal

https://www.academia.edu/36000370/Order_to_Cash_O2C_Cycle_with_Table_details_in_Oracle_Apps

P2P Cycle in Oracle Apps (Step by Step Process and Tables)

Shailvi_0312 - username of API

employee table - emp_id, emp_name, dept_id,salary, d_o_j, mang_id - accn db

e1

mang_id -- 50
emp_id - 100
emp_name - Shailvi
dept_id - 1000
salary - 20000
d_o_j- 10 oct 2022

e2

mang_id -- **
emp_id - 50
emp_name - Puhan
dept_id - 1000
salary - 20000
d_o_j- 10 oct 2022

Print manager name using self join

```
select e2.emp_name from employee e1 self join employee e2 on e1.mang_id = e2.emp_id where
e1.emp_id=100;
```

Department table - dept_id, dept_name, --- total salary

```
select sum(salary), d.dept_id from employee e join department d on
e.dept_id = d.dept_id where dept_name = 'IT'
group by dept_id;
```

Df order

Saturday, January 25, 2025 6:42 PM

```
SELECT TO_CHAR(creation_date, 'MM-DD-YY HH24:mi') daily_pick_status, NVL(order_source,'BULK')
order_source,ORGANIZATION ,
COUNT(DISTINCT ( delivery_id) ) picks_dropped,
COUNT( delivery_id) lines -- 8/30/05
FROM CCWONT.CCW_DELIVERY_Interface
WHERE record_type = 'D'
AND order_source = 'DF'
--And TO_CHAR(creation_date, 'MM-DD-YY HH24:mi') between
--AND order_type = 'DFI'
--AND ORGANIZATION IN('IDC','PDC','FDC','RDC') -- 20-SEP-2005 sg
AND request_id IN (SELECT DISTINCT (request_id)
FROM apps.fnd_concurrent_requests
WHERE concurrent_program_id = 100352 --- 43743 9/12/08 Kumu J. Changed 43743 to 100352
(CCWDSLPKREL)
AND TRUNC(requested_start_date) = TRUNC(SYSDATE))
GROUP BY TO_CHAR(creation_date, 'MM-DD-YY HH24:mi'), NVL(order_source,'BULK'),ORGANIZATION
order by daily_pick_status desc
```

Long running

Saturday, January 25, 2025   6:42 PM

```
WITH running_jobs_qry AS
(SELECT sjq.so_ref1 req_id,
       sjq.so_jobid run_id,
       sjq.so_module job_name,
       to_char(to_date('00:00:00','HH24:MI:SS') +(sysdate - so_job_started), 'HH24')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sysdate - so_job_started), 'MI')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sysdate - so_job_started), 'SS') elapsed_time
FROM    AESCMPROD.so_job_queue sjq
WHERE   sjq.so_status_name  = 'RUNNING'),
hist_jobs_qry AS
(SELECT sjh.so_module job_name,
       sjh.so_job_finished - sjh.so_job_started exec_time
FROM    running_jobs_qry rjq,
       AESCMPROD.so_job_history sjh
WHERE   rjq.job_name = sjh.so_module AND
       sjh.so_status_name = 'FINISHED'),
hist_bkp_jobs_qry AS
(SELECT sjhb.so_module job_name,
       sjhb.so_job_finished - sjhb.so_job_started exec_time
FROM    running_jobs_qry rjq,
       AESCMPROD.so_job_history_backup sjhb
WHERE  rjq.job_name = sjhb.so_module AND
       sjhb.so_status_name = 'FINISHED' AND
       sjhb.so_job_started >= SYSDATE - 30
order by sjhb.so_job_started desc
),
summ_qry AS
(SELECT job_name,
       MIN(exec_time) min_exec,
       AVG(exec_time) avg_exec,
       MAX(exec_time) max_exec
FROM     (SELECT job_name, exec_time  FROM hist_bkp_jobs_qry
       UNION SELECT job_name, exec_time  FROM hist_jobs_qry)
       GROUP BY job_name),
final_values_qry AS
(SELECT rjq.req_id,
       rjq.run_id,
       rjq.job_name,
       rjq.elapsed_time,
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.avg_exec), 'HH24')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.avg_exec), 'MI')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.avg_exec), 'SS') avg_time,
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.min_exec), 'HH24')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.min_exec), 'MI')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.min_exec), 'SS') min_time,
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.max_exec), 'HH24')||':'||
       to_char(to_date('00:00:00','HH24:MI:SS') +(sq.max_exec), 'MI')||':'||
        to_char(to_date('00:00:00','HH24:MI:SS') +(sq.max_exec), 'SS') max_time
FROM    running_jobs_qry rjq,
        summ_qry sq
WHERE   rjq.job_name = sq.job_name
order by rjq.elapsed_time desc)
SELECT *
FROM final_values_qry;
```
has context menu

Alert high disk

Saturday, January 25, 2025 6:44 PM

```
select * from gv$session where sid IN ('3630 ') and serial# IN ('38045 ');

--plug audsid value in "where oracle_session_id = ***1834272468"

select oracle_session_id,request_id from apps.fnd_concurrent_requests where oracle_session_id like
'1908812024';
```

Invalid id

Saturday, January 25, 2025 6:48 PM

```
select * from so_sub_vars
where regexp_like(so_sql_body,'OracleMobilityOMTeam@att.com','i');
```

SQL joins

Sunday, February 9, 2025 2:54 PM

-- You have two tables, employees and departments. The employees table contains information about employees, and the departments table contains information about departments.
--**Write a query to get the employee name and the department name for each employee.**

```
select e.first_name || e.last_name  AS "Name of Employees" , d.department_name from employees e
inner join
departments d on e.department_id = d.department_id;
```

Question:
You have two tables, students and courses. The students table contains information about students, and the courses table contains information about courses. Write a query to get a list of all students and the courses they are enrolled in. If a student is not enrolled in any course, show the **student's name with a NULL** for the course.
Tables:
- students: student_id, student_name
- courses: student_id, course_name

Answer:

```
SELECT students.student_name, courses.course_name
FROM students
LEFT JOIN courses
ON students.student_id = courses.student_id;
```

Question:
You have two tables, orders and products. The orders table contains information about orders placed by customers, and the products table contains details of products. Write a query to get a list of all products and the orders that have been placed for them. If a product hasn't been ordered, still show the **product name with a NULL** for the order information.
Tables:
- orders: order_id, product_id, customer_id
- products: product_id, product_name

Answer:

```
SELECT products.product_name, orders.order_id
FROM products
RIGHT JOIN orders
ON products.product_id = orders.product_id;
```

Question:
You have two tables, employees and salaries. The employees table contains employee details, and the salaries table contains salary information for employees. Write a query to get a list of all employees and their salary details. If an employee doesn't have a salary entry, **show NULL for the salary**, and if there is **no employee linked** to a salary, **show NULL** for the employee information.
Tables:
- employees: employee_id, employee_name
- salaries: employee_id, salary

Answer:

```
SELECT employees.employee_name, salaries.salary
FROM employees
FULL OUTER JOIN salaries
ON employees.employee_id = salaries.employee_id;
```

Join Type	**Use When**
INNER JOIN	You want matching rows from both tables.
LEFT JOIN	You want all rows from the left table, and matching rows from the right table.
RIGHT JOIN	You want all rows from the right table, and matching rows from the left table.
FULL OUTER JOIN	You want all rows from both tables, matching where possible, and NULL where not.
SELF JOIN	You want to join a table to itself (e.g., finding relationships within the same table).
CROSS JOIN	You want the Cartesian product of both tables (all possible combinations).

OIC

Wednesday, February 12, 2025 9:52 AM

Oracle Integration Cloud Service is a complete, secure, but lightweight integration solution that enables us to connect our applications in the cloud. It simplifies connectivity between our applications, and can connect both our applications that live in the cloud and our applications that still live on premises.

Flow -
Create connections -> create integrations -> mapping data (if required) -> activate integrations -> monitoring and error handling.

Trigger Connections

Trigger connection is the entry point for an integration. Trigger connection can't be dropped in any other point in the integration flow other than the first point. Some of the adapters that support trigger connections are: REST, SOAP, File, Oracle Database, Oracle Cloud ERP. However, there are more such adapters that support Trigger connections.

Invoke Connections

Invoke Connection is used for invoking any underlying technology/application. This connection can be used any point in the integration flow. Based on the type of connection there will be a need to configure the Request and Response samples when used in an integration. If the connection is one-way, it will not contain response. Some of the examples are: REST, SOAP, File, FTP, Oracle Database, Oracle Cloud ERP, Salesforce, Workday.

App Driven Orchestration pattern is the multi-step pattern, that allows invoking multiple applications in a single flow, doing a for-each loop, if-else logic, applying complex logic, etc.

Scheduled Orchestration pattern allows us to develop an integration that can run on the pre-defined frequency and can be executed on an Adhoc basis.

What is a lookup in Oracle Integration Cloud?

Lookups are associate values used by one application for a specific field to the values used by other applications for the same field. This provides the capability to map values across vocabularies or systems.

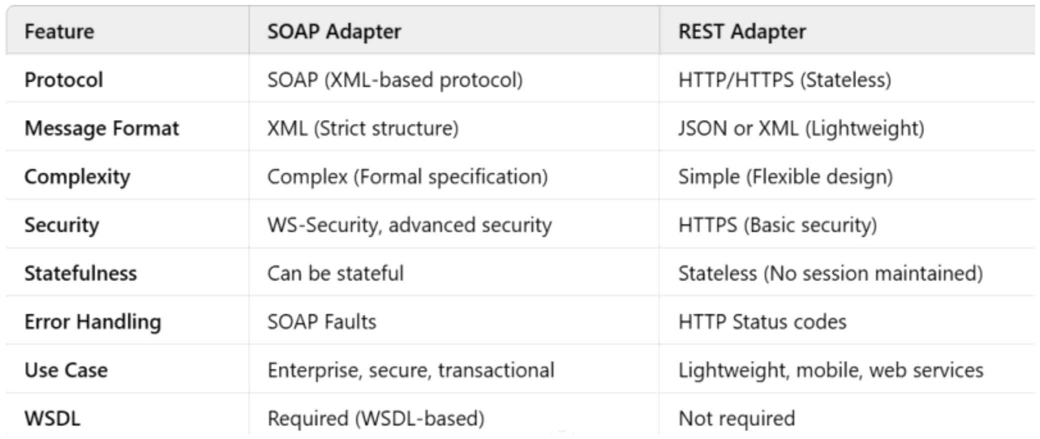
For example, the source application uses a country code to represent the country however target application uses the country name to represent the country. These mapping can be maintained using Lookup and will be referred to by Integration during runtime.

What is a mapper?

A visual mapper is provided that enables us to map fields between applications with different data structures by dragging source fields.

Adapter **allows OIC to connect with various databases, both on-premises and in the cloud.**

SOAP is a protocol while REST is not.



Feature	SOAP Adapter	REST Adapter
Protocol	SOAP (XML-based protocol)	HTTP/HTTPS (Stateless)
Message Format	XML (Strict structure)	JSON or XML (Lightweight)
Complexity	Complex (Formal specification)	Simple (Flexible design)
Security	WS-Security, advanced security	HTTPS (Basic security)
Statefulness	Can be stateful	Stateless (No session maintained)
Error Handling	SOAP Faults	HTTP Status codes
Use Case	Enterprise, secure, transactional	Lightweight, mobile, web services
WSDL	Required (WSDL-based)	Not required

Simple object Access Protocol **Representational State transfer**

When to Use Each:

- **Use SOAP Adapter** when integrating with older or legacy systems that expect strict communication protocols, such as financial systems or enterprise services that need high security and transactional reliability.
- **Use REST Adapter** for modern, scalable, and web-based applications that require lightweight integration with faster performance, especially if the system works with mobile or cloud applications.

Secure File Transfers (SFTP)

- **Use Case**: When you need to ensure that the file transfers are secure, the **SFTP** (Secure FTP) protocol is used, which encrypts the data transmission. This is typically used when dealing with sensitive data.
- **Example**: Sending and receiving financial data files that need encryption during transmission for security and compliance reasons.

Use the **FTP Adapter** in Oracle Integration Cloud when we need to automate the transfer of files between Oracle Integration Cloud and external systems using the FTP/SFTP protocols. It's especially useful when integrating with legacy systems, performing batch processing, or exchanging large files securely and efficiently.

Assign - we can create or update variables.

OIC 2

Tuesday, February 25, 2025 12:50 PM

Basic Questions:

1. **What is Oracle Integration Cloud (OIC)?**
   - **Answer**: Oracle Integration Cloud is a cloud-based integration platform that connects cloud and on-premise applications, automates processes, and handles data exchanges between various systems.
2. **What is an adapter in Oracle Integration Cloud?**
   - **Answer**: An adapter is a pre-built connector that allows OIC to communicate with external systems (e.g., REST, SOAP, FTP, etc.).
3. **What is the difference between an integration and a connection in OIC?**
   - **Answer**: A **connection** defines how OIC communicates with an external system, while an **integration** defines the flow and processing of data between systems.
4. **What are the different types of integrations in OIC?**
   - **Answer**: The main types are **App-Driven Orchestration**, **Scheduled Orchestration**.
5. **What is the role of the REST adapter in OIC?**
   - **Answer**: The REST adapter allows OIC to send and receive data via RESTful web services, supporting HTTP methods like GET, POST, PUT, DELETE.
6. **How does OIC handle error scenarios?**
   - **Answer**: OIC provides built-in error handling, including retries, alerts, and logging for failed integrations.
7. **What is orchestration in OIC?**
   - **Answer**: Orchestration in OIC is a sequence of actions (like invoking web services, transformations, etc.) that automate business processes.
8. **What is the difference between synchronous and asynchronous integrations?**
   - **Answer**: **Synchronous** integrations wait for a response (real-time), while **asynchronous** integrations process in the background (batch or delayed).
9. **What is the FTP adapter used for in OIC?**
   - **Answer**: The FTP adapter is used for file-based integrations to transfer files between OIC and external FTP/SFTP servers.
10. **What is the purpose of the "Visual Builder" in OIC?**
    - **Answer**: Visual Builder allows users to build and customize applications and user interfaces that interact with integrated systems.

Intermediate Questions:

1. **What is the difference between SOAP and REST adapters in OIC?**
   - **Answer**: **SOAP** uses XML and is more rigid with WSDL, while **REST** is simpler, uses JSON, and is more lightweight.
   - **WSDL** stands for **Web Services Description Language**. It is an XML-based language used for describing the functionality of a web service, specifically a **SOAP** web service.
2. **How does OIC ensure data security during integration?**
   - **Answer**: OIC supports security protocols like OAuth, WS-Security, and SSL/TLS to secure data during integration.
3. **What is mapping in OIC?**
   - **Answer**: Mapping is the process of transforming data from one format (e.g., XML) to another (e.g., JSON) during integration.
4. **What is a business event in OIC?**

- **Answer**: A business event is a trigger that initiates an integration process when a specific event occurs in a system.

5. **How do you handle large file transfers in OIC using FTP?**
   - **Answer**: Large files are transferred via FTP/SFTP adapters, with options to split files into chunks or stream data for efficiency.
6. **What is the monitoring feature in OIC?**
   - **Answer**: OIC provides a monitoring dashboard to track integration status, errors, and logs in real-time.
7. **How can you schedule integrations in OIC?**
   - **Answer**: Integrations can be scheduled based on specific time intervals, such as daily, weekly, or at a custom time, using OIC's scheduler.
8. **How does OIC handle data transformation?**
   - **Answer**: OIC supports data transformation using graphical mappings, XSLT, or predefined templates.
9. **What are pre-built integrations in OIC?**
   - **Answer**: Pre-built integrations are ready-to-use templates or connectors for common applications like Oracle ERP, HCM, and third-party services.
10. **What is the use of the Database Adapter in OIC?**
    - **Answer**: The Database Adapter allows OIC to connect to relational databases to read, write, or update data in database tables.

Advanced Questions:

1. **What is the difference between Oracle Integration Cloud and Oracle SOA Cloud Service?**
   - **Answer**: OIC is more lightweight and designed for cloud-to-cloud and cloud-to-on-premise integrations, while SOA Cloud is more complex, supporting large-scale, enterprise-level integrations with advanced features like BPEL and orchestration.
2. **How do you handle versioning in Oracle Integration Cloud?**
   - **Answer**: OIC supports version control, allowing you to manage different versions of integrations and ensuring backward compatibility.
3. **What are "message events" in OIC?**
   - **Answer**: Message events in OIC are triggered when specific messages or files are received, initiating the integration flow.
4. **What are hybrid integrations in OIC?**
   - **Answer**: Hybrid integrations combine cloud and on-premise systems, using OIC’s adapters and agents to integrate disparate systems.
5. **How do you ensure high availability in Oracle Integration Cloud?**
   - **Answer**: High availability is achieved by using OIC’s scalable architecture, redundancy, and failover features to ensure continuous operation.
6. **What are some best practices for handling errors in OIC?**
   - **Answer**: Best practices include setting up retries, defining error handling flows, sending alerts, and logging detailed error information.
7. **How does OIC integrate with Oracle Cloud Applications like ERP or HCM?**
   - **Answer**: OIC uses prebuilt adapters and connectors to integrate with Oracle Cloud applications, facilitating seamless data exchange.
8. **What is a "dead-letter queue" in OIC?**
   - **Answer**: A dead-letter queue holds messages or files that cannot be processed due to errors, allowing for later examination and troubleshooting.
9. **How do you secure data using SFTP in OIC?**
   - **Answer**: SFTP provides secure file transfer over an encrypted channel, ensuring data integrity and

confidentiality during integration.

10. **What is an "error handling framework" in OIC?**
    - **Answer**: An error handling framework defines how errors are caught, logged, retried, and notified to users, ensuring smooth operation during integration.

OIC 3 prac with steps

Wednesday, February 19, 2025 12:27 PM

1. How would you integrate an on-premise system with Oracle Cloud ERP using Oracle Integration Cloud?

Step-by-Step Answer:

Step 1: Set up Connections

- First, create a **connection** to Oracle Cloud ERP using the **Oracle ERP Cloud Adapter**.
- Similarly, create a **connection** to the on-premise system using an appropriate adapter (e.g., **SOAP Adapter** or **FTP Adapter**, depending on the system).

Step 2: Install On-Premise Agent

- To connect the on-premise system to Oracle Cloud Integration, install the **On-Premise Agent**.
- Ensure that the agent has access to the on-premise system and is properly configured in Oracle Integration Cloud.

Step 3: Create an Integration

- Create a new **Orchestration Integration** in OIC.
- For **data flow**, map the on-premise system's data to the Oracle Cloud ERP's expected format.

Step 4: Set Up Error Handling

- Define **error handling** mechanisms such as retry logic, notifications, and logging for failed transactions.

Step 5: Activate and Monitor

- Activate the integration and monitor it using **OIC's monitoring dashboard** to track errors, performance, and success of the integration.

2. How would you implement an integration that reads a file from an FTP server and sends it to a cloud service via REST API?

Step-by-Step Answer:

Step 1: Configure the FTP Adapter

- Create a **FTP Connection** in OIC to connect to the FTP server.
- Configure the **FTP Adapter** to read files (e.g., CSV or XML) from the FTP server.

Step 2: Use Data Mapping

- If the file is in XML or CSV format, use OIC's **mapping tools** to convert the data to the required format (e.g., JSON or XML) to match the target cloud service's REST API.

Step 3: Configure REST API Connection

- Create a **REST connection** in OIC to connect to the cloud service.
- Configure the **REST Adapter** to send data using the appropriate HTTP method (e.g., POST, PUT).

Step 4: Create Integration and Orchestration

- Create an **Orchestration Integration** that reads the file from the FTP server, transforms the data, and sends it to the REST API.
- Define the data flow from FTP to REST.

Step 5: Error Handling and Logging

- Implement error handling for file transfer failures and API request failures.
- Set up **logging** to capture detailed error messages for debugging.

Step 6: Activate and Test

- Activate the integration and test by manually placing files on the FTP server to verify that they are processed and sent to the cloud service.

3. You need to automate the synchronization of customer records between an on-premise system and Oracle Sales Cloud. How would you approach this?

Step-by-Step Answer:

Step 1: Create Connections

- Create a **Sales Cloud Adapter** connection for Oracle Sales Cloud.
- Create a connection for the on-premise system (e.g., **SOAP Adapter**, **Database Adapter**).

Step 2: Set Up the On-Premise Agent

- Install and configure the **On-Premise Agent** to enable communication between OIC and the on-premise system.

Step 3: Create a Scheduled Orchestration

- Create a **Scheduled Orchestration Integration** in OIC that runs at specific intervals (e.g., every night at 2 AM) to synchronize customer records.
- Define the data flow from the on-premise system to Oracle Sales Cloud, making sure to handle any necessary data transformation.

Step 4: Implement Data Mapping

- Use the **mapping tool** to map customer data fields from the on-premise system format to the Oracle Sales Cloud format.

Step 5: Set Up Error Handling and Notifications

- Implement **error handling** to manage any failed synchronization jobs.
- Set up **email notifications** to alert stakeholders about failed or successful integrations.

Step 6: Test and Activate

- Test the integration by running it manually before scheduling it. Ensure that the customer data is transferred correctly to Sales Cloud.
- Activate the scheduled integration.

4. How would you handle a scenario where a file needs to be processed from a partner's SFTP server, but the file is too large for a single transfer?

Step-by-Step Answer:

Step 1: Set Up SFTP Connection

- Create a **SFTP Adapter** connection in OIC to access the partner's SFTP server.
- Configure the SFTP adapter to fetch large files from the partner's server.

Step 2: Use Chunking for Large Files

- Configure **chunking** to break the file into smaller parts for easier transfer. You can either:
    - Split the file into smaller parts and process each part individually.
    - Use **streaming** to handle large files as they are being transferred.

Step 3: Process the File in Batches

- Once the file is transferred in parts or chunks, create a **Batch Processing** integration in OIC.
- Use an **Orchestration** to handle the file processing in smaller pieces, thus avoiding memory overload.

Step 4: Error Handling and Logging

- Implement **retry logic** in case a chunk fails to process.
- Set up **logging** to capture the success or failure of each file chunk to ensure proper tracking.

Step 5: Send Processed File

- Once the file is processed, you can send the transformed data to the appropriate system (e.g., Oracle ERP or CRM) using the **ERP Adapter** or **REST Adapter**.

5. How would you create an integration to synchronize data from Oracle ERP Cloud to an external database using Oracle Integration Cloud?

Step-by-Step Answer:

Step 1: Create Connections

- Create a connection to **Oracle ERP Cloud** using the **ERP Adapter** in OIC.
- Create a connection to the **external database** (e.g., Oracle Database) using the **Database Adapter**.

Step 2: Define the Data Flow

- Use the **ERP Adapter** to query data from Oracle ERP Cloud (e.g., customers, orders, products).
- Use the **Database Adapter** to insert or update records in the external database.

Step 3: Create Integration

- Create a **Basic Routing Integration** or an **Orchestration Integration** in OIC to fetch data from ERP Cloud and push it to the external database.
- Define the required operations (e.g., SELECT from ERP Cloud and INSERT/UPDATE in the database).

Step 4: Implement Data Mapping

- Map the data fields between Oracle ERP Cloud's format and the external database schema using OIC's **mapping tools**.

Step 5: Error Handling and Validation

- Implement **data validation** to ensure the data is correctly mapped.
- Set up **error handling** for issues such as database connection failures or data format issues.

Step 6: Test and Activate

- Test the integration to ensure data synchronization works correctly.
- Activate the integration and monitor it using the **monitoring dashboard** in OIC.

6. How would you set up an integration that listens for new orders placed on an eCommerce platform (via REST API) and automatically updates Oracle Sales Cloud?

Step-by-Step Answer:

Step 1: Configure the REST Connection

- Create a **REST Adapter** connection in OIC to listen for incoming HTTP requests from the eCommerce platform.
- Set the **REST endpoint** URL in OIC to receive new orders.

Step 2: Define the REST Operation

- Define the **GET or POST operation** for receiving order data (order ID, customer details, items, etc.) from the eCommerce platform.

Step 3: Set Up Sales Cloud Connection

- Create a **Sales Cloud Adapter** connection in OIC to update customer orders in Oracle Sales Cloud.

Step 4: Create an Orchestration Integration

- Create an **Orchestration Integration** that:
    1. Receives the order data from the eCommerce platform (via REST API).
    2. Maps the incoming order data into the format required by Sales Cloud.
    3. Uses the **Sales Cloud Adapter** to send the order data to Oracle Sales Cloud.

Step 5: Implement Error Handling and Logging

- Set up **error handling** for failed API calls (e.g., invalid data, network issues).
- Set up **logging** for tracking order processing.

Step 6: Test and Activate

- Test the integration by placing a test order on the eCommerce platform and ensuring it is correctly updated in Oracle Sales Cloud.
- Activate the integration for production use.

OIC prac

Tuesday, February 25, 2025 12:55 PM

1. How would you integrate an on-premise application with a cloud-based Oracle ERP system using Oracle Integration Cloud?

Expected Answer:

- Discuss using the **Oracle ERP adapter** in OIC to connect the cloud ERP system and how you would configure the **on-premise agent** to handle on-premise data.
- Highlight using **pre-built integrations** or custom orchestration for transferring data between the on-premise system and the Oracle ERP system.
- Emphasize handling network/firewall configurations and security protocols like **OAuth** for authentication.

2. In Oracle Integration Cloud, how would you handle a scenario where a file is received through FTP and needs to be processed and sent to a cloud application?

Expected Answer:

- Use the **FTP adapter** to fetch files from the external FTP server.
- Apply **data transformation** using OIC's **mapping tools** (e.g., XSLT, graphical mapping) to transform the file into the appropriate format required by the cloud application.
- Use the **REST or SOAP adapter** to send the transformed data to the cloud application.
- Include **error handling** mechanisms like retries and notifications in case of failures.

3. You are tasked with creating an integration that synchronizes customer data between an on-premise legacy system and Oracle Sales Cloud. How would you approach this?

Expected Answer:

-
- Design a **scheduled orchestration** integration that runs at predefined intervals to synchronize customer data.
- Perform **data transformation** between the legacy system's format (e.g., flat files, XML) and the format required by Oracle Sales Cloud.
- Implement **error handling** and **logging** to monitor synchronization jobs.

4. How would you implement an integration between Oracle Integration Cloud and a third-party REST API service?

Expected Answer:

- Use the **REST adapter** to connect to the third-party REST API service.
- Define the required **HTTP methods** (GET, POST, PUT) based on the API documentation.
- Handle **authentication** (e.g., OAuth, Basic Auth) and include **headers/parameters** as per the API specification.
- Use OIC's **mapping** capabilities to transform data between Oracle Cloud and the third-party service, ensuring the correct request and response formats.
- Implement **error handling** for possible API failures or timeouts.

5. How would you create an integration that listens for new orders placed on an eCommerce website (via a REST API) and sends order details to Oracle ERP for processing?

Expected Answer:

- Use the **REST adapter** in OIC to listen for **HTTP requests** from the eCommerce website.
- Map the incoming order data (e.g., order ID, customer details, products) into a format suitable for Oracle ERP.
- Use the **ERP adapter** to send the order data to the Oracle ERP system for processing.
- Set up **error handling**, such as retries and logging for failed integrations, and ensure **data validation** to ensure that the data is properly formatted.

6. How can you implement security in an integration between Oracle Integration Cloud and a third-party service?

Expected Answer:

- Implement **OAuth 2.0** or **Basic Authentication** for secure access to the third-party service.
- Use **HTTPS** for encrypted data transmission.
- If integrating with SOAP services, configure **WS-Security** for message-level security, including signature and encryption.
- Discuss securing the **Oracle Integration Cloud instance** using **IP whitelisting** and **role-based access control**.

7. In a scenario where multiple services need to be invoked sequentially within an Oracle Integration Cloud orchestration, how would you implement this?

Expected Answer:

- Create an **orchestration integration** in OIC that invokes services sequentially by defining a series of **invoke activities**.
- Ensure proper **data mapping** between service invocations to pass data from one service to the next.
- Use **variables** in the orchestration to store interim results and pass them between service calls.
- Implement **error handling** to ensure that if one service fails, the subsequent invocations are either skipped or retried.

8. How would you use Oracle Integration Cloud to transform XML data into a CSV file and send it via FTP to a partner's server?

Expected Answer:

- Use the **XML to CSV transformation** in OIC by creating a mapping between XML elements and CSV fields.
- After transforming the data, use the **FTP adapter** to send the resulting CSV file to the partner's FTP server.
- Optionally, set up **error handling** to log issues if the FTP upload fails or if transformation issues occur.

9. What steps would you take to monitor and troubleshoot an integration in Oracle Integration Cloud that is not functioning as expected?

Expected Answer:

- **Check the OIC monitoring dashboard** for detailed logs and error messages related to the integration.
- Review the **integration flow** and verify the connections to ensure they are properly configured.
- Look for specific **error codes** or **exception messages** and use them to pinpoint issues.
- If the integration involves third-party systems, verify their availability and check any **API limits** or **authentication issues**.
- Use **retry logic** and examine **tracking and logging** features for more granular details of where the integration might be failing.

10. How would you handle a situation where a file needs to be processed from a partner's SFTP server, but the file is too large for a single transfer?

Expected Answer:

- Use the **SFTP adapter** to retrieve the file in **chunks**. OIC can process files in smaller segments if the file is too large for a single transfer.
- Configure the **chunking mechanism** (either in the adapter or within the integration logic).
- Optionally, you can use **streaming** to process the file in real-time as it's being transferred.
- Set up **error handling** to manage potential interruptions or failures during the transfer of large files.

11. How would you set up a schedule for an integration in Oracle Integration Cloud that runs every night at 2 AM?

Expected Answer:

- In OIC, set up the integration as a **scheduled orchestration**.
- Specify the frequency (e.g., daily) and the time (e.g., 2 AM) in the **scheduler** settings.
- Ensure that the integration handles retries in case of failure, and logs success/failure for audit purposes.

12. If you have to integrate two systems that use different data formats (e.g., one uses XML and the other uses JSON), how would you handle the transformation?

Expected Answer:

- Use **OIC's mapping capabilities** to transform the data from **XML to JSON** (or vice versa).
- Create a **data mapping** within the integration design to convert data elements from one format to another.
- Ensure the mapping handles any data type conversions, missing fields, or discrepancies in structure.

SCM (O2C)

Friday, March 7, 2025 11:40 AM

The **Order-to-Cash (O2C)** cycle is a fundamental business process that encompasses everything from receiving an order to collecting payment. It is essential for managing the flow of goods and services and ensuring the company gets paid for those goods or services.

Step 1: Order Entry

This is the initial stage of the O2C cycle where a customer places an order, and the details are entered into the Oracle system. This creates records in two important tables: the **OE_ORDER_HEADERS_ALL** table and the **OE_ORDER_LINES_ALL** table.

Key Tables in Oracle Apps:

1. **OE_ORDER_HEADERS_ALL**:
    - This table stores **header** information for each sales order. The header is like the "main" or "overall" record that holds general details about the order.
    - **Important columns** (fields) in this table:
        - **HEADER_ID**: A unique ID automatically assigned to the order. Think of it as the "ID number" for the order.
        - **ORG_ID**: The organization (or business unit) that is handling the order.
        - **ORDER_NUMBER**: A unique number assigned to the order.
        - **SHIP_FROM_ORG_ID**: The ID of the organization (or warehouse) that will ship the order.
        - **SHIP_TO_ORG_ID**: The ID of the organization (or address) where the order will be delivered.
        - **FLOW_STATUS_CODE**: The status of the order (for example, 'Entered' means the order has been entered into the system but not yet processed).
    - **Sample Query:**

      ```
      SELECT HEADER_ID, ORG_ID, ORDER_TYPE_ID, FLOW_STATUS_CODE, TRANSACTIONAL_CURR_CODE, SHIPPING_METHOD_CODE, SHIP_FROM_ORG_ID,
      SHIP_TO_ORG_ID
      FROM OE_ORDER_HEADERS_ALL
      WHERE ORDER_NUMBER = 66405;
      ```
        - This query helps you find all details about a specific order (with order number 66405 in this case).
2. **OE_ORDER_LINES_ALL**:
    - This table stores **line-level** details for the items in the order. Each "line" represents a specific product or service that the customer has ordered.
    - **Important columns** in this table:
        - **LINE_ID**: A unique ID for each line item in the order.
        - **HEADER_ID**: This links the line item to the corresponding header (order) in the **OE_ORDER_HEADERS_ALL** table.
        - **ORDERED_ITEM**: The name or ID of the item that has been ordered.
        - **INVENTORY_ITEM_ID**: The unique ID of the item in the company's inventory system.
        - **PRICING_QUANTITY**: The quantity for which pricing is applied.
        - **ORDERED_QUANTITY**: The total quantity of the item ordered.
        - **FLOW_STATUS_CODE**: The status of the line item (for example, 'Entered' means the line item has been added but not yet processed).
        - **UNIT_SELLING_PRICE_PER_PQTY**: The price of one unit of the item.
    - **Sample Query:**

      ```
      SELECT LINE_ID
      FROM OE_ORDER_LINES_ALL
      WHERE HEADER_ID = 190452;
      ```
        - This query helps you find all the line items for a specific order (based on the HEADER_ID).

          ```
          SELECT ORDERED_ITEM, INVENTORY_ITEM_ID, PRICING_QUANTITY, ORDERED_QUANTITY, FLOW_STATUS_CODE, UNIT_SELLING_PRICE_PER_PQTY
          FROM OE_ORDER_LINES_ALL
          WHERE LINE_ID = 388401;
          ```
        - This query helps you find all details of a specific line item (with LINE_ID 388401).

In Simple Words:

- **Order Entry** is the first step where a customer's order is recorded in the system.
- Two main pieces of data are captured:
    - **Order Header**: General information like the order number, who's buying it, where it's being shipped from, and the current status of the order.
    - **Order Lines**: Details of the specific items ordered (like what the customer is buying, how much of each item, and the price).

By storing this data in these tables (**OE_ORDER_HEADERS_ALL** for the order itself and **OE_ORDER_LINES_ALL** for the items in the order), the company can track and process the order in subsequent stages of the O2C cycle.
This way, whenever you need to check the details about a customer's order or its individual items, you can simply query these tables to retrieve the necessary information.

Step 2: Order Booking

After you enter the order in the system (which we saw in the **Order Entry** stage), the next step is **Order Booking**. **Order Booking** means that the order is finalized and confirmed, making it eligible for processing and shipment.

What Happens During Order Booking?

- **Order Entry is Complete**: When the customer's order has been successfully entered into the system and is confirmed to be valid (with pricing and quantities checked), the order can be moved to the next stage.
- **Booking the Order**: When you "book" the order in Oracle Apps, you finalize it, confirming that it's ready to be processed and shipped. This is done by clicking the "Book Order" button.
- **Status Changes**:
    - **OE_ORDER_HEADERS_ALL** table: The FLOW_STATUS_CODE in this table changes from **'Entered'** (when the order was initially created) to **'BOOKED'** (after it is

confirmed).
  - **OE_ORDER_LINES_ALL** table: The status for each line item changes from **'Entered'** to **'AWAITING_SHIPPING'**, meaning the item is ready to be shipped.
  - **WSH_DELIVERY_DETAILS** table: The status changes to **'R' (Ready to release)**, meaning the items are ready to be processed for shipping.

Key Table for Order Booking:

1. **WSH_DELIVERY_DETAILS**:
   - This table stores **delivery details** for the order. It keeps track of the specifics of what needs to be shipped to the customer.
   - **Important columns** in this table:
     - **DELIVERY_DETAIL_ID**: A unique ID generated for each delivery record. It links to the order header and line.
     - **SOURCE_HEADER_ID**: This is the HEADER_ID from the **OE_ORDER_HEADERS_ALL** table. It ties the delivery record to the specific sales order.
     - **SOURCE_LINE_ID**: This is the LINE_ID from the **OE_ORDER_LINES_ALL** table, which links the delivery to specific items in the order.
     - **RELEASED_STATUS**: This indicates whether the delivery is ready to be processed for shipment. If it is **'R'**, it means the order is **ready to release** (prepare for shipping).
     - **SOURCE_CODE**: Indicates the source of the order (like the sales order).
     - **CUSTOMER_ID**: The unique ID for the customer who placed the order.
     - **INVENTORY_ITEM_ID**: The ID of the product being shipped.
     - **SHIP_FROM_LOCATION_ID**: The ID of the location (warehouse) from which the goods will be shipped.
     - **SHIP_TO_LOCATION_ID**: The ID of the location (address) where the goods will be delivered.
     - **REQUESTED_QUANTITY**: The amount of the item the customer requested.
     - **SHIPPED_QUANTITY**: The amount of the item actually shipped (can be different from requested quantity if there's a partial shipment).
     - **SUBINVENTORY**: The inventory location where the items are stored.
     - **SHIP_METHOD_CODE**: The method of shipment, such as ground or air shipping.

Example Query:

The sample query provided is used to find the details of all delivery records related to a specific order:

```
 SELECT DELIVERY_DETAIL_ID,
      SOURCE_HEADER_ID,
      SOURCE_LINE_ID,
      SOURCE_CODE,
      CUSTOMER_ID,
      INVENTORY_ITEM_ID,
      ITEM_DESCRIPTION,
      SHIP_FROM_LOCATION_ID,
      SHIP_TO_LOCATION_ID,
      MOVE_ORDER_LINE_ID,
      REQUESTED_QUANTITY,
      SHIPPED_QUANTITY,
      SUBINVENTORY,
      RELEASED_STATUS,
      SHIP_METHOD_CODE,
      CARRIER_ID
FROM WSH_DELIVERY_DETAILS
WHERE SOURCE_HEADER_ID = 190452;
```

- **What this query does**: It retrieves all details about the delivery associated with the sales order (identified by SOURCE_HEADER_ID = 190452). This includes information about the customer, items being shipped, requested and shipped quantities, shipping method, and more.

In Simple Words:

- **Order Booking** is when the system confirms that the order is ready to move forward and be processed for shipment.
- Once you "book" the order, the order status in the system changes:
  - The order is marked as **'BOOKED'**.
  - The individual line items are marked as **'AWAITING_SHIPPING'**, meaning they're waiting to be shipped.
  - The delivery details are created, and the status of the shipment is marked as **'Ready to Release'** (indicating that the goods are ready to be prepared for shipping).

The **WSH_DELIVERY_DETAILS** table is crucial here as it tracks all the details about the products being shipped, where they're coming from, and where the y're going.

Step 3: Launch Pick Release

Pick Release is the process where the items that were ordered are taken out of the warehouse inventory and made ready for shipment. Essent ially, it's the action of allocating the available inventory to fulfill the customer's order and preparing it for shipment.
Here's how it works:

1. **Inventory Allocation**: The system checks the warehouse to see if the ordered items are available in stock. If they are, the system allocates these items to the specific order.
2. **Movement to Staging Area**: Once the inventory is allocated, the items are moved from the general warehouse area to a specific **shipping staging area**. This is where the items are kept before they are packed and shipped.
3. **Warehouse Notification**: Pick release also informs the warehouse personnel that it's time to start picking and moving the items for shipping.

Important Tables:

1. **OE_ORDER_LINES_ALL**:
   - In this table, each item of the sales order is tracked, and its status will be updated during pick release.
   - **FLOW_STATUS_CODE**: This column tracks the status of each item in the order. After pick release:
     - The **status changes to 'PICKED'** when the item has been picked from the inventory.
     - The status may also be **'AWAITING_SHIPPING'** if the item is still waiting to be shipped (depends on the "Auto Pick Confirm" setting).

2. **WSH_DELIVERY_DETAILS**:
   - This table stores details about the delivery of each order.
   - **RELEASED_STATUS**: This column tracks the status of the delivery. After the pick release process:
     - The **status will be 'S' (Submitted for Release)** if the item is ready for shipment but hasn't been confirmed.
     - The **status will be 'Y' (Pick Confirmed)** when the items have been picked and are confirmed as ready for shipment (again, this depends on settings like "Auto Pick Confirm").
3. **WSH_DELIVERY_ASSIGNMENTS**:
   - This table tracks the assignments for the delivery.
   - It links the **delivery details** from the **WSH_DELIVERY_DETAILS** table to the actual physical task of moving the items.
   - **DELIVERY_ID**: This is the unique ID of the delivery and is linked to **DELIVERY_DETAIL_ID** in the **WSH_DELIVERY_DETAILS** table, which helps track the specific item being delivered.

Sample Code:

1. **First Query:**

```
SELECT DELIVERY_DETAIL_ID
FROM WSH_DELIVERY_DETAILS
WHERE SOURCE_HEADER_ID = 190452;
```

   - This query retrieves the **DELIVERY_DETAIL_ID** for a specific order, identified by **SOURCE_HEADER_ID** (190452). It helps to identify all the items associated with this specific order that need to be shipped.
2. **Second Query:**

```
SELECT DELIVERY_ASSIGNMENT_ID,
     DELIVERY_ID,
     PARENT_DELIVERY_ID,
     DELIVERY_DETAIL_ID,
     PARENT_DELIVERY_DETAIL_ID,
     CREATION_DATE,
     CREATED_BY,
     LAST_UPDATE_DATE,
     LAST_UPDATED_BY,
     ACTIVE_FLAG,
     TYPE
FROM WSH_DELIVERY_ASSIGNMENTS
WHERE DELIVERY_DETAIL_ID = 3966467;
```

   - This query fetches the details from the **WSH_DELIVERY_ASSIGNMENTS** table, which tracks the assignments for a particular delivery (in this case, the one identified by **DELIVERY_DETAIL_ID = 3966467**).
   - It returns details like the **DELIVERY_ASSIGNMENT_ID** (unique task ID for this delivery), **DELIVERY_ID**, the **PARENT_DELIVERY_ID** (if applicable), and other relevant tracking information.

In Simple Words:

- **Pick Release** is the process where the warehouse picks the products from the inventory that are needed to fulfill the customer's order and moves them to a specific staging area for shipping.
- Once the items are picked, the **status** of the order and the items changes in the system:
  - In the **OE_ORDER_LINES_ALL** table, the item's status is marked as **'PICKED'** (if the item is picked) or **'AWAITING_SHIPPING'** (if it's still waiting).
  - In the **WSH_DELIVERY_DETAILS** table, the **RELEASED_STATUS** changes to **'S'** (Submitted for Release) or **'Y'** (Pick Confirmed) depending on the settings in the system.
- The **WSH_DELIVERY_ASSIGNMENTS** table helps track which specific tasks (like picking or moving) are assigned to warehouse personnel for the order.

This process ensures that the warehouse is properly informed about which items need to be picked and moved to the shipping ar ea, getting the items ready to be sent out to the customer.

Step 4: Ship Confirm the Order

Ship Confirm is the final step in the shipping process where the items in the staging area are physically shipped to the customer. Once th e shipment is completed, this action updates the system to notify that the order is now fully shipped, and it also adjusts inventory levels accordingly.
Here's what happens during the **Ship Confirm** process:

1. **Shipping the Items**: The items that were previously moved to the **shipping staging area** are now physically sent to the customer.
2. **System Notification**: By performing the Ship Confirm process, the system is notified that the shipment has been completed. This means the order is considered "shipped," and the system updates the inventory to reflect that these items are no longer available for sale.
3. **Updating Inventory**: The inventory system will automatically update to show that the items have been shipped and are no longer in stock.
4. **Reports Triggered**: When you confirm the shipment, it will also trigger the generation of some important shipping and billing documents:
   - **Interface Trip Stop**: This is used to stop the trip and confirm that the goods have been delivered.
   - **Packing Slip Report**: This document lists the items that were shipped, including quantities and descriptions.
   - **Bill of Lading**: This is a legal document that serves as a receipt for the shipped goods and can be used in case of disputes or claims.
   - **Commercial Invoice**: This document is required for international shipments and lists the details of the goods being shipped, including their value and customs information.

Key Tables Affected:

1. **OE_ORDER_LINES_ALL**:
   - This table keeps track of each item in the sales order. When the items are shipped, the status in this table is updated.
   - **FLOW_STATUS_CODE**: This field will be updated to **'SHIPPED'** when the item has been successfully shipped to the customer.
2. **WSH_DELIVERY_DETAILS**:
   - This table tracks the delivery details for each item in the order.
   - **RELEASED_STATUS**: This field will be updated to **'C' (Shipped)** to indicate that the order has been shipped to the customer.

In Simple Words:

- **Ship Confirm** is the process where the physical shipment of the items is confirmed in the system, meaning the items are on their way to the customer.
- After the shipment is confirmed:
    - The **OE_ORDER_LINES_ALL** table updates to show that the item has been **'SHIPPED'**.
    - The **WSH_DELIVERY_DETAILS** table updates the **RELEASED_STATUS** to **'C'** (which stands for "Shipped").
- At the same time, several important documents are generated:
    - **Packing slip** (which lists the shipped items),
    - **Bill of Lading** (a receipt for the shipment),
    - **Commercial Invoice** (for international shipments).

By confirming the shipment, you notify the system that the order is complete, and the inventory is adjusted accordingly. The customer is also notified that their goods have been shipped.

Step 5: Creating Invoices in Receivables (O2C Cycle):

In the **Order to Cash (O2C)** cycle, once the goods are shipped, you need to create invoices in the **Receivables** system. This means you're creating a document that will request payment from the customer for the products or services they've purchased.

Workflow Background Process:

To create invoices automatically, Oracle runs something called the **Workflow Background Process**. This process checks the shipping records (which contain the information about the goods shipped to the customer) and transfers the relevant data into the Receivables system. Essentially , it moves the data from one system (Sales Order) to the next system (Receivables) to generate invoices.

RA_INTERFACE_LINES_ALL Table:

This table temporarily stores the data that is transferred into the Receivables system before the invoice is actually created . Some important fields (columns) in this table are:

1. **INTERFACE_LINE_ID**: This is a unique identifier for the line of the order. It links to the **OE_ORDER_LINES_ALL** table, where the sales order details are stored.
2. **INTERFACE_LINE_CONTEXT**: This tells what type of transaction this record is related to (e.g., "Order Entry").
3. **INTERFACE_LINE_ATTRIBUTE1**: It stores the **Order Number**, which uniquely identifies the specific order.
4. **INTERFACE_LINE_ATTRIBUTE3**: It stores the **Delivery ID**, which refers to the specific delivery of goods to the customer.

Sample SQL Code for RA_INTERFACE_LINES_ALL:

```
SELECT INTERFACE_LINE_CONTEXT,
      INTERFACE_LINE_ATTRIBUTE1,
      INTERFACE_LINE_ATTRIBUTE3
FROM RA_INTERFACE_LINES_ALL
WHERE INTERFACE_LINE_ID = 388401;
```

This query retrieves the **context**, **order number**, and **delivery ID** for a specific line of the order.

RA_CUSTOMER_TRX_ALL Table:

Once the Workflow Background Process has transferred the necessary information, the actual invoice gets created and stored in the **RA_CUSTOMER_TRX_ALL** table. This table holds the **invoice header information**.

Some important columns here:

1. **INTERFACE_HEADER_ATTRIBUTE1**: This column stores the **Order Number** again, which helps in identifying the original sales order associated with the invoice.
2. **INTERFACE_HEADER_ATTRIBUTE2**: This column contains the **Order Type** (such as "Standard Order").
3. **TRX_NUMBER**: This is the **Invoice Number** generated for this transaction. It's the unique identifier for the invoice itself.
4. **CUSTOMER_TRX_ID**: This is a unique ID for the invoice (the transaction record in the Receivables system).
5. **COMPLETE_FLAG**: This flag indicates whether the transaction is completed or not.
6. **SHIP_DATE_ACTUAL**: This is the **actual shipping date** when the goods were shipped to the customer.

Sample SQL Code for RA_CUSTOMER_TRX_ALL:

```
SELECT INTERFACE_HEADER_ATTRIBUTE2,
      CUSTOMER_TRX_ID,
      TRX_NUMBER,
      CUST_TRX_TYPE_ID,
      COMPLETE_FLAG,
      SHIP_DATE_ACTUAL
FROM RA_CUSTOMER_TRX_ALL
WHERE INTERFACE_HEADER_ATTRIBUTE1 = '66405';
```

This query retrieves the **order type**, **invoice number**, **customer transaction ID**, and other invoice details for a specific order number.

RA_CUSTOMER_TRX_LINES_ALL Table:

This table stores the **line items** of the invoice (the individual items or services that were sold to the customer). The details about each product in the invoi ce are recorded here.

Key columns include:

1. **INTERFACE_LINE_ATTRIBUTE1**: The **Order Number** for reference.
2. **INTERFACE_LINE_ATTRIBUTE2**: The **Order Type** (type of order, like "Standard").
3. **INTERFACE_LINE_ATTRIBUTE3**: The **Delivery ID** (links to the specific delivery made to the customer).
4. **INTERFACE_LINE_ATTRIBUTE4**: The **Waybill** (shipping document number).
5. **INTERFACE_LINE_ATTRIBUTE5**: The number of items in this invoice line (e.g., how many units of the product).
6. **INTERFACE_LINE_ATTRIBUTE6**: The **Line ID** (unique identifier for the invoice line item).
7. **INTERFACE_LINE_ATTRIBUTE7**: **Picking Line ID** (relates to the picking process in the warehouse).
8. **INTERFACE_LINE_ATTRIBUTE8**: **Bill of Lading** (document for shipping).
9. **INTERFACE_LINE_ATTRIBUTE9**: **Customer Item Part** (unique identifier for the item sold).
10. **INTERFACE_LINE_ATTRIBUTE10**: **Warehouse** (where the product came from).
11. **INTERFACE_LINE_ATTRIBUTE11**: **Price Adjustment** (if there was any discount or price change).

12. **INTERFACE_LINE_ATTRIBUTE12**: **Shipment Number** (identifies the shipment associated with this invoice).
13. **INTERFACE_LINE_ATTRIBUTE13**: **Option Number** (if the product had different options, like size or color).
14. **INTERFACE_LINE_ATTRIBUTE14**: **Service Number** (if the transaction involves services).

Conclusion:

In simple terms, the process is:

- When the goods are shipped, a record of that shipping information is created and passed to the **Receivables** system.
- The **Workflow Background Process** helps transfer the necessary details (like order number, delivery ID) from shipping to receivables.
- The **RA_INTERFACE_LINES_ALL** table stores this data temporarily.
- The **RA_CUSTOMER_TRX_ALL** table holds the main invoice header details, and the **RA_CUSTOMER_TRX_LINES_ALL** table holds the detailed line items of the invoice.
- Once the invoice is generated, it’s available in the Receivables system for further processing like payment collection.

The workflow ensures that the sales order, shipment, and invoice creation are all properly connected and tracked throughout t he process.

Step 6: Create Receipt in Receivables:

Once you have created invoices and shipped the goods to the customer, the next step in the **Order to Cash (O2C) cycle** is to receive the **payment** from the customer. This is done by creating a **receipt** in the **Receivables** system.

Underlying Table: AR_CASH_RECEIPTS_ALL

When a receipt (payment) is created in the system, the information is stored in the **AR_CASH_RECEIPTS_ALL** table. This table keeps track of all the **payments (receipts)** made by the customer.

Key points to understand about this table:

- **CASH_RECEIPT_ID**: This is a **unique system-generated ID** that identifies each receipt (payment) made by the customer. Think of it as a unique reference number for each payment.
- **FLOW_STATUS_CODE in OE_ORDER_LINES_ALL**:
    - The **FLOW_STATUS_CODE** in the **OE_ORDER_LINES_ALL** table is updated to **‘CLOSED’** once the receipt (payment) is successfully processed.
    - This means that the payment has been received, and the order is considered fully paid or "closed," meaning no further actionis needed for this particular order.

In Simple Terms:

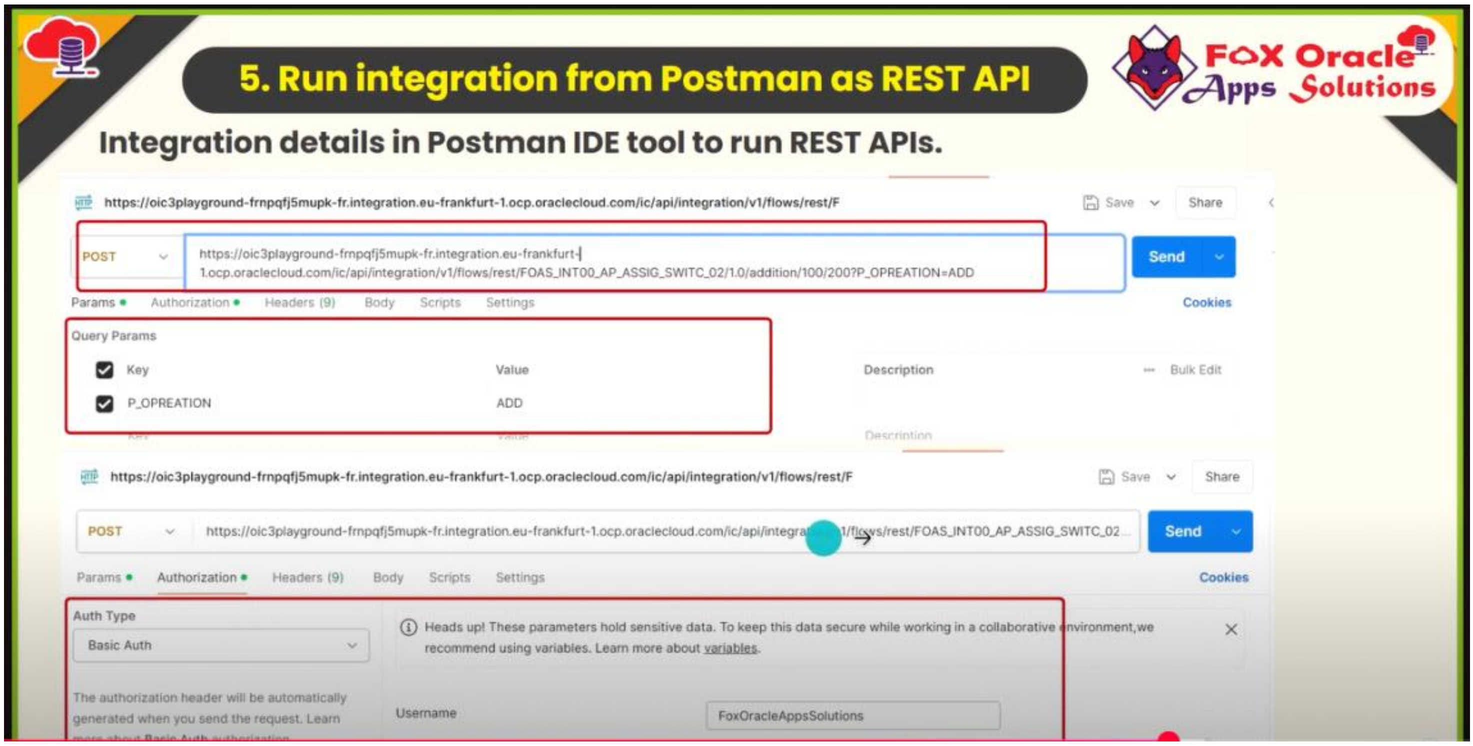
1. **Payment Receipt**: Once a customer makes a payment, the company records this payment in the system.
2. **AR_CASH_RECEIPTS_ALL**: This table stores the details of the payment, and each payment gets a unique **CASH_RECEIPT_ID**.
3. **Order Status Update**: Once the payment is recorded, the status of the corresponding sales order (in **OE_ORDER_LINES_ALL**) is marked as **‘CLOSED’**, indicating that the order has been fully paid for.

So, when a receipt is created:

- The payment is recorded in **AR_CASH_RECEIPTS_ALL** with a unique ID.
- The sales order is updated to **‘CLOSED’**, meaning it’s fully paid and no further actions are needed.

- **O2C (Order to Cash)**: Focuses on the company's process of **selling** goods or services to customers and collecting payments.
- **P2P (Procure to Pay)**: Focuses on the company's process of **buying** goods or services from suppliers and paying for them.

Monday, March 17, 2025 12:10 PM



5. Run integration from Postman as REST API

Integration details in Postman IDE tool to run REST APIs.

https://oic3playground-frnpqfj5mupk-fr.integration.eu-frankfurt-1.ocp.oraclecloud.com/ic/api/integration/v1/flows/rest/F

POST https://oic3playground-frnpqfj5mupk-fr.integration.eu-frankfurt-1.ocp.oraclecloud.com/ic/api/integration/v1/flows/rest/FOAS_INT00_AP_ASSIG_SWITC_02/1.0/addition/100/200?P_OPREATION=ADD

Query Params

Key	Value	Description
✓ P_OPREATION	ADD	

POST https://oic3playground-frnpqfj5mupk-fr.integration.eu-frankfurt-1.ocp.oraclecloud.com/ic/api/integra... 1/flows/rest/FOAS_INT00_AP_ASSIG_SWITC_02...

Auth Type: Basic Auth

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.

The authorization header will be automatically generated when you send the request. Learn

Username: FoxOracleAppsSolutions

https://oic98863535-ocuocictrng12-ld.integration.uk-london-1.ocp.oraclecloud.com/ic/api/integration/v1/flows/rest/APP_INT_02_ASSIGN_SWITCH/1.0/addition/%7Bnum_1%7D/%7Bnum_2%7D?P_Operation=[P_Operation-value]

SQL

Monday, April 15, 2024 1:54 PM

Table -

create table customer (Customer_ID int, Customer_Name varchar(20), Address varchar(100), City varchar(20),country varchar(15), salary number(20));

insert into customer values(1,'Arun','Katra Bazar','Chapra','India',10000);
insert into customer values(2,'Rupa','Katra Bazar','Chapra','India',13000);
insert into customer values(3,'Tilak','Kala Pipal','Indore','India',14000);
insert into customer values(4,'Shailvi','120 Hanover Sq.','London','UK',15000);
insert into customer values(5,'Shriya','Obere Str. 57','Berlin','Germany',16000);
insert into customer values(6,'Ruchi','Karol bagh','Nagpur','India',18000);
insert into customer values(7,'Abhira','120 Hanover Sq.','London','UK',13000);
insert into customer values(8,'Swamy','Fauntleroy Circus','London','UK',15000);
insert into customer values(9,'Samiksha','Hauptstr. 29','Bern','Switzerland',17000);
insert into customer values(10,'Samiksha','Via Monte Bianco 34','Torino','Italy',16000);
insert into customer values(11,'Simpi','2732 Baker Blvd.','Eugene','Usa',19000);
insert into customer values(12,'Simran','City Center Plaza 516 Main St.','Elgin','USA',18000);
insert into customer values(13,'Ramesh','Garden House Crowther Way ','Cowes','UK',14000);
insert into customer values(14,'Tushar Kant','1900 Oak St. ','Vancouver','Canada',12000);
insert into customer values(5,'Shriya','Obere Str. 57','Berlin','Germany',29000);

delete from customer where CUSTOMER_ID not in (select min(CUSTOMER_ID) from customer group by CUSTOMER_NAME,city);

Where clause using NOT

SELECT * FROM Customers where not country = 'Germany' and city = 'london' ;

AND

OR

want to return all customers from Germany but also those from Spain:
SELECT * FROM Customers
WHERE Country = 'Germany' OR Country = 'Spain' ;

Between operator -

To filter the records in the specified range
SELECT * FROM Customers where CustomerId between 10 and 20;
SELECT * FROM Customers where CustomerId >= 10 and CustomerId < 20 ;

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;
```
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

Not between -

SELECT * FROM Customers where CustomerId not between 10 and 20;
SELECT * FROM Customers where not CustomerId >= 10 and CustomerId < 20 ;
SELECT * FROM Customers where CustomerId < 10 or CustomerId <20;

ORDER By clause - To order the retrieved records in ascending or descending order of the provided column data

SELECT * FROM Products **order by price ;**
SELECT * FROM Customers order by customerName desc, country asc;

Using between operator with text
SELECT * FROM Customers
WHERE Country **between 'Mexico' and 'UK'** order by country;

SELECT * FROM Customers
WHERE Country **not between 'Mexico' and 'UK**' order by country;

IN operator - Simplify the process of providing multiple values to the same column name.
SELECT * FROM Customers
WHERE Country = 'Germany' or country = 'Canada' or country = 'France' order by country;
to overcome these many OR - we will use IN operator instead of many OR

```
SELECT * FROM Customers
WHERE Country in ('Germany','Canada','France') order by country;

SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK') order by country;
```

Like Operator and Wildcard characters
LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
-- if we want to use wildcard characters then we must use Like operator
Purpose : Pattern matching
wildcard characters

1. **% - multiple chars**

```
SELECT * FROM Customers
WHERE CustomerName like 'a%';

SELECT * FROM Customers
where country like 'g%y';

SELECT * FROM Customers
WHERE CustomerName LIKE 'a%' OR CustomerName LIKE 'b%';
```

2. **_ - single char**

```
SELECT * FROM Customers
where country like 'Ca_ada';

SELECT * FROM Customers
where country like 'u__';

SELECT * FROM Customers
where country like 'fIN_A%';
```

3. [] wildcard returns a result if *any* of the characters inside gets a match.

```
SELECT * FROM Customers
WHERE CustomerName LIKE '[bsp]%';
```

4. - wildcard allows you to specify a range of characters inside the [] wildcard.

```
SELECT * FROM Customers
WHERE CustomerName LIKE '[a-f]%';
```

Aliases in place of table column name - using as is optional

```
SELECT CategoryID as ID,categoryName as Name FROM Categories  ;
```

Using Aliases With a Space Character

```
SELECT ProductName AS [My Great Products]
FROM Products;

SELECT ProductName AS "My Great Products"
FROM Products;
```

Concatenate columns -

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address
FROM Customers;

SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address
FROM Customers;

SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;
```

Limit keyword - to improve the performance while retrieving the records from the table having huge number of records
How many records needs to be displayed on the page at a time.

```
SELECT * FROM Customers LIMIT 3;
SELECT * FROM Customers LIMIT 3,6;
```

Breaking the SQL statements into multiple lines - when sql statement is lengthy then understanding will not easy to overcome this problem we can break into multiple lines

```
SELECT CustomerID,CustomerName,ContactName,City
FROM Customers
where country = "France"
limit 3;
```

Built- in Functions and their categories -
Different RDBMS software - MYSQL,oracle
MYSQL - Categories of built-in function in mysql
Date and Time Functions

```
SELECT current_date();
select curdate();
select current_time();
select curtime();
select now(); - give date + time
Select sysdate() - give date + time
select year('2022-04-22');
select day('2022-04-22');
select monthname('2022-04-22');
SELECT * FROM Orders where month(orderDate) = 7;
```

Aggregate Functions

```
SELECT max(Price) FROM Products;
SELECT count(Price) FROM Products;
SELECT min(Price) FROM Products;
SELECT avg(Price) FROM Products;
```

String Functions

UPPER() - String category in MYSQL
Convert the text under table column to uppercase

```
SELECT UPPER("SQL Tutorial is FUN!") AS UppercaseText;
SELECT upper(CustomerName) as name FROM Customers;
```

LOWER() - String category in MYSQL
Convert the text under table column to lowercase

```
SELECT lower(CustomerName) as name FROM Customers;
```

Length() - to find the length of the column data
Ex - arun - length is 4 SELECT length('arun') ;

```
SELECT country, length(country) as length FROM Customers;
SELECT country, length(country) as length FROM Customers where length(country)>5;
```

Instr() - to find the position of the text under column name

```
SELECT country,instr(country,'er') as position FROM Customers;
```

Substr() - to find the portion of text from the data of the specified column

```
SELECT country,substr(country,2,4) as substring  FROM Customers;
```

Concat() - to add 2 or more column data together

```
SELECT CONCAT('W3Schools', '.com');
SELECT concat(address,' ',postalCode) as [Full address] FROM Customers;
```

Trim() - to remove the leading and trailing spaces in the column data ' Arun ' -> 'Arun'

```
SELECT TRIM('     SQL Tutorial     ') AS TrimmedString;
```

Numeric Functions

Abs() - retrieve absolute value (positive value) irrespective of whether the given number is either positive number or negative.

```
SELECT ABS(-243.5); -> 243.5 o/p
```

Mod() - will return the remainder value of the numeric data of specified column

```
SELECT MOD(18, 4); -> 2 - remainder
SELECT quantity, mod(quantity,3) FROM OrderDetails;
```

Greatest() and least() -

```
SELECT GREATEST("w3Schools.com", "microsoft.com", "apple.com");
```

Truncate() - return the numerical value with the specified number of digits after the decimal point.

Ex- truncate(123.456,1) = o/p -> 123.4
SELECT price, truncate(price,1) FROM Products;

Power() -
SELECT POWER(4, 2); = 4*4 = 16

Sqrt() -
SELECT SQRT(64); = 8

Arithmetic operators -

1. Addition
   a. select 5+4;
   b. SELECT price,price+10 FROM Products;
2. Subtraction
   a. select 5-4;
3. Multiplication
   a. select 5*4;
4. Division
   a. select 5/4;
5. Modulus
   a. select 5%4;

Creating, deleting, viewing and using databases.

Create database test;
For deleting - drop database test;
If you want to use database particularly then --- "USE" keyword we can use - use database name;
For viewing database - show database
For viewing the existing tables in DB - show tables;

Creating Table -
Create table employee(id int,name varchar(20),experience int)

Creating Table using existing table-
create table emp as
   Select id,name from employee;

Describe tablename - **all details comes of the table**

How to delete tables -
Drop table tablename - drop table emp;

Inserting data into the tables -

select * from employee;
describe employee;
insert into employee values(1,'Arun',3);
insert into employee values(2,'Varun',5);
insert into employee values(3,'Gita',6);
insert into employee(id,name) values(4,'Sita');
insert into employee values(1,'Arun',null);

Data Types in SQL -
1. String - Varchar
2. Numeric - int, double, boolean
3. Date and Time - date, time, datetime, year

Null Value, Is Null Operator, Is Not Null Operator

If particular column doesn't contains any value then it will consider as Null Value

select * from emp where name is null;
select * from emp where name is not null;

Delete Statements for deleting the Table records

For deleting database & Tables - drop database
For deleting records - delete
    delete from emp where id=2;

Update statements along with SET keyword

update emp
set name = 'Kiran'
where id = 2;

update emp set name = 'Siman' -- all records will update on their own

Rename statement along with TO keyword -- for renaming the table name

rename table emp to employees;

Alter Statement, Add, Modify column, Rename column and drop column

alter table emp add location varchar(12) - add 1 column
alter table emp
modify column id varchar(12); - it will modify the datatype of particular column
alter table emp
rename column location to LOC; - it will rename the column name
alter table emp
drop column loc - to delete column

SET AutoCommit

If we use set autocommit =0 --> records wont saved permanently (used for temporary change)
set autocommit = 1 --> records saved permanently

If autocommit is 0 then we can manually commit the records by using 'commit' command

Rollback - Revert the temporary changes done on a particular table

Truncate - no matter set autocommit = 0 - it always delete records permanently

Single/Multiple line comments - "--" -> for single line
"/* ……… */ -> for multiple line

Group by Clause - To group the retrieved records according the specified column (select is used with Where clause)
GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

	select count(name) AS "Name of Employees" ,experience from emp group by experience ;

Having Clause - if we want to add conditions in group by clause (having is used with group by clause)

select count(name) AS "Name of Employees" ,experience from emp group by experience having experience > 2;	

Sequence of where,having,group by, order by clause --
Where>group by > having> order by

select count(name) AS "Name of Employees" ,experience from emp id where id>1 group by experience having experience > 2; - doubt

SET Operators - if we want to retrieve records of 2 tables together

Types of SET operators -

1. **UNION(duplicates eliminate- check all columns -if all are same only then it will remove)**

   Rules -
   - Select statements of each table must retrieve the same number of columns
   - Columns provided in select statement should have the similar data type and in the same order to work properly

2. **UNION ALL(won't ignore duplicates)**

3. **INTERSECT(common)**
4. **MINUS - usually 1st table records comes whatever mention firstly (t1 - t2 = t1)/(t2-t1 = t2)**

select * from emp union select * from emp2;	select * from emp union all select * from emp2	select * from emp intersect select * from emp2	select * from emp minus select * from emp2

Tables and Aliases
select c.id,c.salary,p.item_id,p.price from customer c, products p where c.id=p.item_id;

Joins -- To join different tables

Prerequisties for the tables to get joined --
Those table we want to join should have common column

Types -

1. **Inner Join (Equi Join or simple join)**

Common records retrieve

select * from customer
inner join
products on customer.id = products.item_id;

2. **Left join (Left outer join)**

   1st table is left & 2nd table is right - retrieve common + all left table records and in place of other column of 2nd table it will give null

   select * from customer
   left join
   products on customer.id = products.item_id;

3. **Right join (Right outer join)**

   1st table is left & 2nd table is right - retrieve common + all right table records and in place of other column of 1st table it will give null

   select * from customer
   left join
   products on customer.id = products.item_id;

4. **Full join (Full outer join) - both tables joins full -- if any values not matched then it will give null values instead of**

   select * from customer
   full join
   products on customer.id = products.item_id;

5. **Self join - same table 2 times - here it will compare dept_id and id if they are same it will retrieve**

   select * from customer c,customer o where c.id=o.dept_id;

Sub Query -

Two types of Sub Query -

1. **Single Row sub query - resulting in only one record sub query (EXISTS operator can be used in single row sub query, IN,ANY,ALL can't used)**

   i. Find all customers who belongs to Hari's city
   select * from customer where city = (select city from customer where name = 'Hari');
   ii. Find the 2nd maximum price of products
   select max(price) from products where price < (select max(price) from products)
   iii. Find 3rd max(price)
   select max(price) from products where price < (select max(price) from products where price < (select max(price) from products))
   iv. Find least price sold
   select * from products where price = (select min(price) from products)

   Select id,salary from emp e1 where N-1 = (select count(distinct salary) from emp e2 where e2.salary > e1.salary);

   Select min(salary) from (select distinct salary from emp order by salary desc) where rownum<=2; -- 2nd highest salary

   Select * from(select ename,salary,dense_rank() over(order by salary desc)rank from emp) where rank= &num;

2. **Multiple Row sub query - resulting in multiple records sub query (we can use IN,ALL,ANY,EXISTS)**

IN Operator - mainly used in multiple row sub query

Without IN operator we can only give one value in the where clause condition

Select * from custome where country = 'USA'
Select * from customers where country In ('USA','UK','Italy')
SELECT * FROM Customers
WHERE Country not IN ('Germany', 'France', 'UK');

Find the different products which fall into the specific categories.

SELECT * FROM Products where categoryID IN (SELECT categoryID FROM Categories where categoryName like 'C%');

ANY / ALL Operator

operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

SELECT * FROM Products where ProductID < any (SELECT ProductID FROM OrderDetails where quantity = 1);
SELECT * FROM Products where ProductID < all (SELECT ProductID FROM OrderDetails where quantity = 1);

EXISTS Operator

SELECT * FROM Orders where exists (SELECT * FROM Customers where customerID>89); after exists giving true condition so query will give whole orders table

Delete duplicate rows from the table

select customer.*,rowid from customer ;
delete from customer where rowid not in (select min(rowid) from customer group by customer_name)

Using sub queries for retrieving the records from multiple tables - internal subqueries should be one to one (big - country) relation

Integrity Constraints - condition that can be applied on column data

Types - Not null, Unique, Primary Key, Foreign Key, check, default

Not Null -

create table employees(id int not null, name varchar(10), experience int);
insert into employees(name,experience) values('Arun',12);
insert into employees(id,name) values(2,'Arun');
insert into employees(name,experience) values('Arun',13);/ insert into employees(id,name) values(null,'Arun'); -- it will give error and not to be inserted

Unique - we can't insert same values whose unique column

create table orders (**id int unique**, name varchar(20)) / create table emp (id int, name varchar(12),experience int, ***unique(id))*** ;
insert into orders values (1,'Arun');
insert into orders values (1,'Arun'); -- it will give error and not to be inserted

create table emp (id int, name varchar(12),experience int, **unique(id, name)**)
insert into emp values (1,'Arun',12);
insert into emp values (1,'Arun',13); -- it will give error and not to be inserted
insert into emp values (1,'Varun',12); -- it will work because uniqueness will work together .. If anyone is different records will be inserted.

Primary Key - Not null + Unique -- we need to give any values to that column.

create table emp1(id int primary key,name varchar(15),experience int); / create table emp (id int, name varchar(12),experience int,primary key(id));
insert into emp1 values(1,'Arun',12);
insert into emp1 values(1,'VArun',14); --- it will give error and not to be inserted
insert into emp1(name,experience) values('Tharun',11);--- it will give error and not to be inserted

create table emp (id int, name varchar(12),experience int,primary key(id,name));
insert into emp(name,experience) values('Arun',12);--- it will give error and not to be inserted
insert into emp(id,name) values(1,'Avrun');

Foreign Key -

Common column should be common data type but name can differ it is fine.

2 table - one is parent table having primary key & other is child table having foreign key.
Child table is going to depend on parent table.

Ex - parent table - employee , child table - salary

Rule 1 - Data inserted into the child table should have the same data of the common column of parent table.

Rule 2 - you can't delete the records in the parent table having the associated records in the child table.

ON DELETE CASCADE

Rule 3 - 1st delete child table records then parent table records.

-- Parent table --(Reference table) -- primary key
create table employee(id int primary key, name varchar(15),experience int);

insert into employee values(1,'Arun',12);
insert into employee values(2,'varun',5);
insert into employee values(3,'tharun',13);

-- Child table -- Foreign Key
create table salary(id int, sal int, foreign key(id) references employee(id)); /
create table salary(id int, sal int, foreign key(id) references employee(id)) on delete cascade; - want to delete from both table associated records.

insert into salary values(1,120000);
insert into salary values(9,820000); -- it won't be inserted because '9' is not there in the parent table

Check integrity constraints -

create table empone(id int,name varchar(15), experience int check (experience >5));
insert into empone values(1,'Arun',12);

insert into empone values(2,'varun',1); - it won't be inserted because experience is not >5

create table emptwo(id int,country varchar(15) check(country in ('India','USA','UK')));

Default integrity constraints -

Table column - if no value is inserted into that column - the given/specified default value will be inserted into that.

create table empthree(id int,experience int default 5);
insert into empthree(id) values (4);-- experience will take as by default 5

Auto_increment - to increase the value of specified table column by 1
create table empone(id int primary key auto_increment ,name varchar(15)); -- without primary key auto_increment wont work.
-- if we wont give any value to primary key column it will auto increment by 1 and if we give value it will take that value.
-- in starting of giving value we wont give it will take by default as 1 (we can set the default value is auto_increment = 100).

Insert Into - Copy the records from one table to another table

describe newcustomer;
create table newcustomer(ncustomer_id int, cname char(35), caddress varchar(200),ccity varchar(35),ccountry varchar(20)); -- no need to mention all columns of older table
insert into newcustomer select * from customer;
select * from newcustomer; -- but here instead of * mention those name which are mentioned above (it can be less but not more) in newly created table.

create table newcustomer(ncustomer_id int, cname char(35), caddress varchar(200),ccity varchar(35),ccountry varchar(20));
insert into newcustomer select * from customer where country = 'UK';

AS Keyword - To create a new table with the records of an existing table

create table newcustomers
as
select * from customer;
select * from newcustomers;
	create table newcustomerss
	as
	select * from customer;
	select customer_name, city from newcustomers;

IfNull() - select (ifnull(salary,0/15)+1000) from emp;

Case,when,then,End

SELECT ProductName,Price, CASE when Price < 10 then 'The price is less than 10' when Price = 10 then 'The Price is equal to 10' when Price > 10 then 'The Price is greater than 10' END as Price_Details,Unit FROM Products;	SELECT * FROM Customers order by (Case when country in ('USA','UK') then Country when country not in ('USA','UK') then city end);	SELECT * FROM Customers where city = (Case when country in ('USA','UK') then City when country not in ('USA','UK') then 'Berlin' end);

Delimiter - Default Delimiter in SQL is semicolon (;) - want to replace ; then use delimiter

Delimiter //
select * from customer//

Views
Every time we need small change in the existing table, we need to create a new table.
Using views, we can overcome the need for creating more tables even though the required columns and data are available in the existing table.

create view customerview
as
select CUSTOMER_ID,CUSTOMER_NAME,CITY,COUNTRY from customer;

select * from customerview;

If we change anything in real table it will reflect in view and if we change in view then it will reflect in real table also.

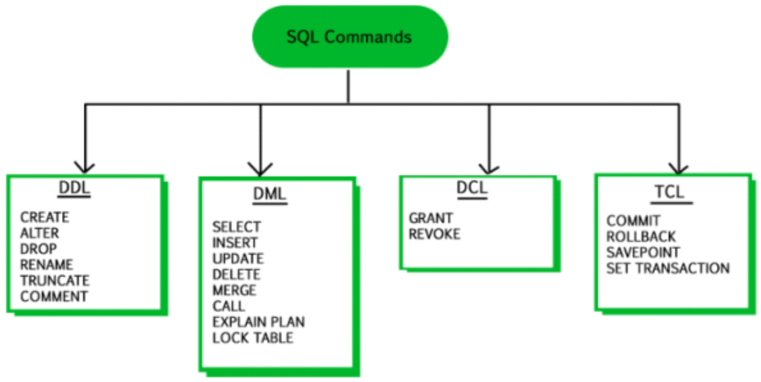
Indexes -
We can create indexes for the tables.
It will improve the performance i.e. We can retrieve the records at a faster speed from the tables.
The tables having indexes, we can retrieve the records in a faster way
Insertion and updating will be slow

select * from customer where country = 'India';

```
create index country_index
                on
customer(country);
show indexes from customer;
select * from customer where country = 'India';
```

SQL Statements Types -

1. **DQL - Data Query Language (*select* statement)**
2. **DML - Dara Manipulation Language**
3. **DDL - Data Definition Language**
4. **TCL - Transaction Control Language**
5. **DCL - Data Control Language**



DRL (retrieval) - select

Delete -- data udd gya but table and structure avail h
Drop -- data and table structure (dono udd gya)
Truncate -- pehle drop+delete (phr new table same nam ki bnagega jisme data ni rhega)

Grant and Revoke - (DCL)

Temporary tables - There may be some situations where the application has to create a table for a purpose and thereafter using the table it has delete it.

Inserting Null values -

1. **Directly -- insert into empone values(1,Null);**
2. **No value -- if you are not providing while inserting then it will take as null**

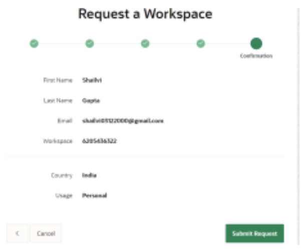
Using Trim() -
select trim('a' from 'arun') from customer;

Wild Cards as normal characters -
_ (single), %(many) -select * from customer where customer_name like 'A___';

Database objects - the things we can create under any database using the create statements in SQL are nothing but the database objects.

Optimized Query -
Query should be efficient

```
select
   concat(first_name, ' ', last_name) AS 'patient_name',
   ROUND(height / 30.48, 1) as 'height "Feet"',
   ROUND(weight * 2.205, 0) AS 'weight "Pounds"', birth_date,
CASE
        WHEN gender = 'M' THEN 'MALE'
  ELSE 'FEMALE'
END AS 'gender_type'
from patients
```



Request a Workspace

Confirmation

First Name Shalvi

Last Name Gupta

Email shalvi09122000@gmail.com

Workspace 6285436322

Country India

Usage Personal

SELECT
  COUNT(*) AS patients_in_group,
  FLOOR(weight / 10) * 10 AS weight_group
FROM patients
GROUP BY weight_group
ORDER BY weight_group DESC;
	SELECT
	  TRUNCATE(weight, -1) AS weight_group,
	  count(*)
	FROM patients
	GROUP BY weight_group
	ORDER BY weight_group DESC;
		SELECT
		  count(patient_id),
		  weight - weight % 10 AS weight_group
		FROM patients
		GROUP BY weight_group
		ORDER BY weight_group DESC
			select count(has_insurance),sum(admission_cost) as admission_total
			from
			(
			   select patient_id,
			   case when patient_id % 2 = 0 then 'Yes' else 'No' end as has_insurance,
			   case when patient_id % 2 = 0 then 10 else 50 end as admission_cost
			   from admissions
			)
			group by has_insurance

OIC Stage File(Write)

Friday, March 21, 2025 12:19 PM

Req : - Create an integration to get employee details as request payload and generate a CSV file based on this employee data and send over the mail.

Sol:
Create App-Driven Integration using REST Adapter having trigger connection.
Stage File - to create the file
Notification - send an email

===== Create a request JSON Payload ====

```
{
	"RequestData":
[
{
	"Name" : "Alex",
	"DOB"  : "01-01-2000",
	"Role" : "IT-Security",
	"DOJ"  : "05-11-2024"
},

{	  "Name" : "Alex",
	"DOB"  : "01-01-2000",
	"Role" : "IT-Security",
	"DOJ"  : "05-11-2024"
}
]
}
```

Now we will create sample file so that OIC take the reference of that sample file to create CSV file.

Create CSV sample file so that OIC will take this--
```
Name, DOB,Role,DOJ,CreationDate,CreatedBy,IntegrationName
Alex,19-03-2025,IT,19-03-2025,19-03-2025,XYZ,XYZ
```

== Response Payload ===

```
{
	"Status" : "Success/Error",
	"FileName" : "xyz.csv"
}
```

--------- Create App driven
Give name - Stage_DEMO_01
Package name - abc
Add the trigger connection which we already created now configure --
Endpoint name - StartRest
Enpoint's URI - /generateFile
Operation - POST
Check both option of configure a request payload and receive the response - next
Drag the JSON Request Payload
Drag the JSON Response Payload in configure Response page
Continue and finish

For creating the file --

It is better to create variable

Assign it after Trigger conn
Name - default assign
Var_FileName = 'XX_EMP_001.csv' -- to generate the file we use this name

For creating the file in OIC we have Stage File (Write File)

Want to call ur action? - WriteFile_Stage

Choose stage file operation - Write File
File name - drag var_FileName
O/p directory - '/temp'
Continue --
Drag the sample file
Enter the record name - RN
Enter the recordSet Name - RSN
Rest option we can keep the same
Continue and finish
Then click on MAP to WRITE_FILE_STAGE
Request Wrapper to response map them
Created Date -- use current_date function by creating a target node
OIC instance ID
Created by - - Integration metadata details - Runtime details- Invoked by
Integration name -Integration metadata details - Integration name

Validate it and go back

Now the stage file that is created it will return file reference in repsonse - file is already created and they will give link of the file

Add Notification before Map of startrest

Add the details in notification
TO - 'shailvi03122000@gmail.com'
Cc -
From - 'no-reply@oracle.com'
Subject - 'Stage File Learning'
Body -
Hi Team,
Please find the attached CSV file which contains employee details.
File Name : {p_FileName} --should be in braces

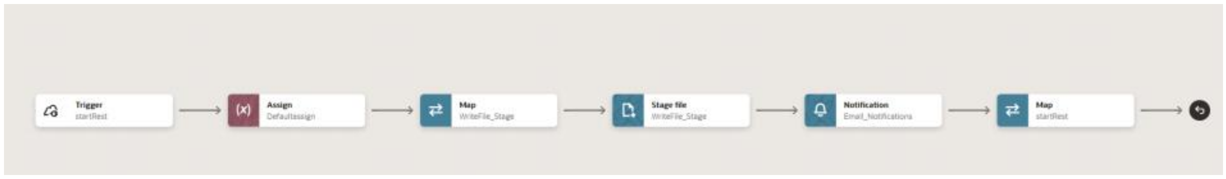
Regards,
OIC Support Team

Parameters - if we want to add File name too in the body so we can add parameter -- we cant do this in body so we are creating here just for notif only
Parameter naeme - P_File name drag value -- variable - var_File name
Attachments - Give value --- under WriteFile_stage --take File reference under ICS file.
Request one is -- mapper ignore this

INT_01_ST
AGE_DE...
Goto MAP startRest --
Provide response wrapper
Status - 'Success'
File Name - var_FileName

Remove business identifier

Save it , activate it and run it.



OIC (FTP)

Friday, March 21, 2025 1:18 PM

-- For creating the FTP/sFTP connection
Connection - create - select FTP adapter

Give name of conn - XX_FTP_CONN
Role - Invoke
Click on create

Then configuration page will open provide the details
Host Address -
Server Port-
Security Policy - FTP server access Policy (choose this)
Username -
Password -

If you want to make sFTP conn - goto optional properties and choose 'YES'

Test and save this connection

--- Create Integration which will get the file from FILE SERVER
 - **create APP DRIVEN in such a way that while running the integration we will provide the file directory & file name based on that it will download the file from FILE SERVER, it will read the file data and it will return the data in response of this integration.**

For creating the integration --

== Request Payload ===

```
{  "FileName" :"",
   "SourceFileDirectory" : ""
}
```

== Response Payload == want all the column

```
{ "CustomerData" :
        [
        {"PartyNumber" : "",
        "OrganizationName" : "",
        "PartyUsageCode" : "",
        "CurrencyCode" : "",
        "RawPhoneNumber" : "",
        "EmailAddress" : "",
        "Address" :""
        }  ,

        {"PartyNumber" : "",
        "OrganizationName" : "",
        "PartyUsageCode" : "",
        "CurrencyCode" : "",
        "RawPhoneNumber" : "",
        "EmailAddress" : "",
        "Address" :""}
        ]
}
```

---> Create - App Driven - give name (FTP_INT_01)
Add connection which we made earlier (rest trigger)
Give endpoint name - startRest
Endpoint URI - /FTP01
Method - POST
Check the button configure a request payload for this endpoint and configure this endpoint to receive the response.

Continue - configure request page -- paste above request payload
Then - configure response page - paste above response payload
Click continue and Finish

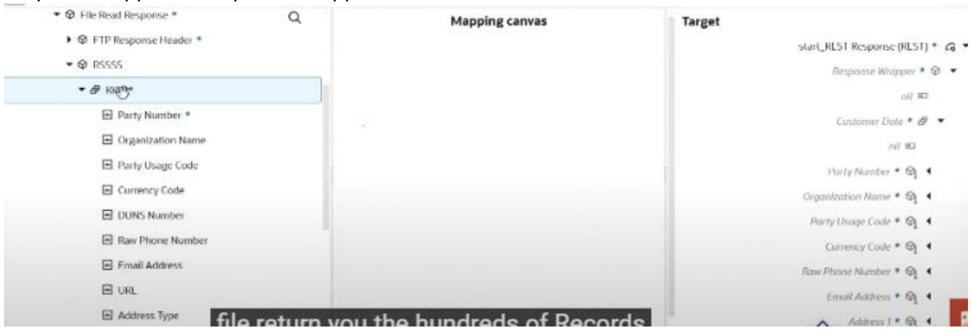
File size is small so instead of downloading we can directly read the content of the file.
For reading the file - use FTP conn
Give endpoint name - ReadFile_FTP
Select operations - Read a File
Input directory -- using mapper we can give the value at that time by overriding so we can provide anything like '/'
File name -- *
Choose csv and continue
Give sample file -- customer_sample file (having sample file with less record)

Give record set name and record name whatever you want
Auto mapper will be in canvas -- readFile_FTP

Map Filename to File name
Directory to sourceFileDirectory

Map to start_Rest

Request wrapper to Response wrapper



It automatically creates FOR EACH LOOP
We can concat city,country,postal code to address by creating a target node.

Validate it and save the integration and remove business identifier

Activate it and run it

While running give filename -- copy from fileserver that file name and
Sourcedirectory -- copy from fileserver

Then run it data will return as response payload

--> To move a file from one place to another in file server
--first remove then place

After the read file invoke endpoint add one FTP connection
Give endpoint name - ArchiveFile_FTP
Continue
Select 'MOVE A FILE'
Directory path - from where we have to pick the file
File Name -
Target Directory path -
Target File name - want to give

We can provide these values while on mapping so just leave these options and continue and finish it.

Click on map of archiveFile
-- we can give the directory in request payload
Add one column
== Request Payload ===

{ "FileName" :"",

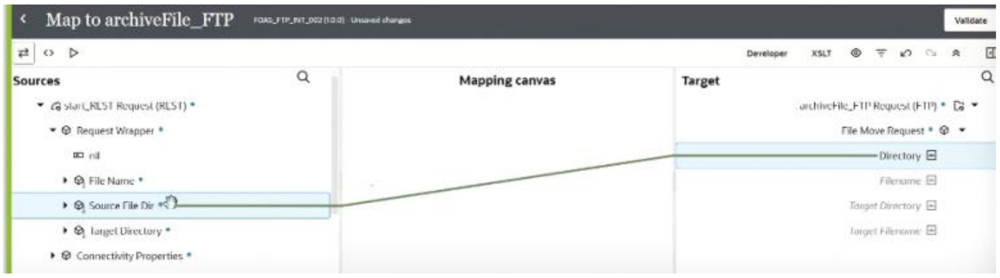
```
    "SourceFileDirectory" : "",
        "TargerDirectory" : ""
}
```

Want to update the simple request payload open rest trigger end point

Just click on prev request payload and paste this RP on inline.

Save it

Open mapper



1) List down all the files present at source file directory
2) Read the all file in OIC
3) Create new file and merger data of all above file in single file
4) Delete the all previously file

-- In this case we don't want to get any response from integration so will create schedule integration (asynchoronus) so that we don't want to give any parameters to this integraiton.

-- create Schedule integration - name as SC_FTP_INT_01
-- we have to read all the file from file server so we will create one variable for this -- add assign to create the variable
Name - default_Assign
Click on + -

var_Source Directory give value the path of that directory

Var_Merge_directory give value the path of new merge directory

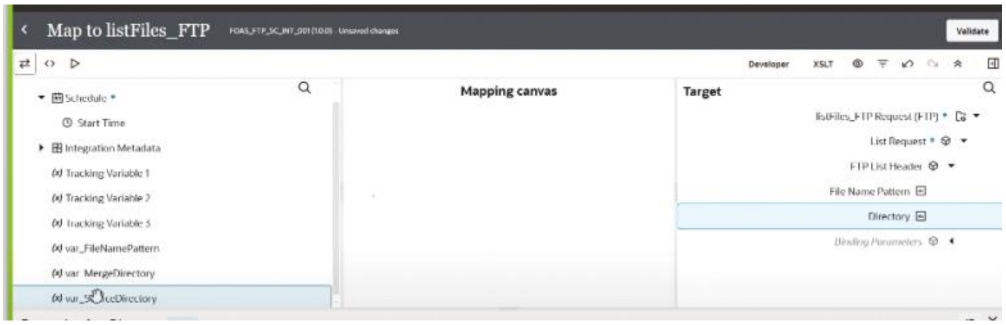
Var_MergeFileName -> 'XX_CUSTOMER_MERGER.CSV'

-- we have to get the filename to read the file
Add 1 more var
Var_FileNamePattern - value - 'XX_CUSTOMER*.csv'

To get the file name -- USE FTP CONNECTION AND LIST FILEs operation choose this

Name - LISTFILES_FTP
Input Directory --> / -- we can give while on mapping it will override
FileName pattern -- *.csv --- we can give while on mapping it will override
Max Files - 10

Open ListFILES_FTP Mapper -



-- for reading the files - choose download the file
We can add the condition to do download when file is more than 1

Add switch
-- give name - check_count
-- list_Files_FTP drag this as a response of this end point expand it
File list -- File -- choose itemCount add >0
Then save it

Add otherwise case -- just add the logger to provide the message only
'File not found'

Add For each loop (if more than 1 file is there to download it)
Repeating element - File under list
Current element name - lustFile_forEach_loopVar--> give this var name
Add FTP conn
Name it - download_FTP
Choose - download
Directory - '/temp' (OIC virtual directory)

Reading the file from OIC directory we have stage file --

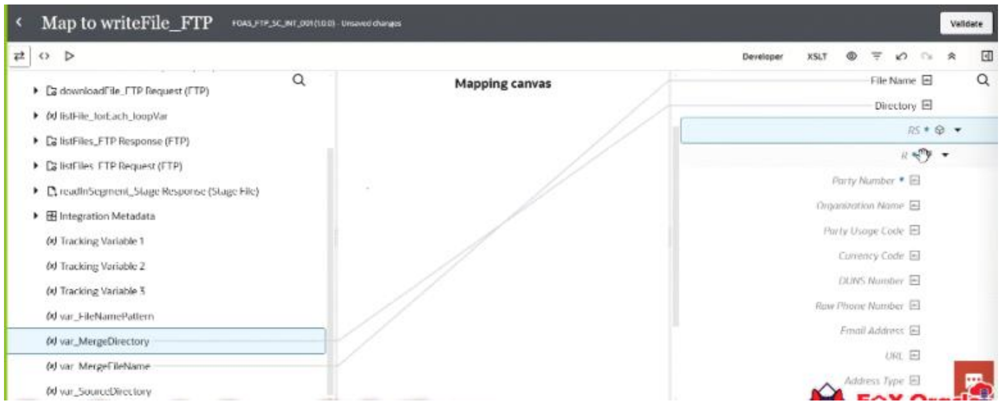
Give action name - READ_FILE
We will choose 'Read File in segments' because it may be possible it will be above 100 MB.
Give file refernece - choose DownloadResponse - file reference.
Continue
Drag sample file
Give record name , record set name
Added stage file scope

Now we have to write the files in merge folders
Choose FTP conn in stage file scope
Action name - write_File_FTP
Output directory - /
File Name pattern - *
Check this append button
Click continue

Drag sample file
Give record name , record set name

Continue and save it

Open WriteFile_FTP



OIC Project

Monday, March 24, 2025 11:09 AM

<u>Use Oracle Integration and ERP Cloud, Import bulk data services with File Based Data Import (FBDI) complaint files.</u>

design and develop File-based Data Integration (FBDI) Import

The goal is to import ERP data such as account payable invoices using processes in ERP Cloud.

The typical flow of this use case is:

1. The user uploads an FBDI based account payable invoice file to an FTP Server.
2. Oracle Integration imports the account payable invoice file into the ERP Cloud.

Sol ---
Use FTP and ERP cloud adapter

1. Read files from an SFTP Server

2. Synchronize account payable invoices into the ERP Cloud

- Enable File Server
- Configure File Server
- Connect to File Server using FTP Client

For Oracle Integration -

- Create File Server connection using FTP adapter
- Create ERP Cloud connection using ERP Cloud adapter

Oracle Integration Cloud

Case Study

You are an Oracle Integration (OIC) integration developer tasked with automating a data exchange process between two systems: **System A** and **System B**.

System A generates daily CSV files containing transaction records, and these files need to be processed by **System B** for further analysis and reporting. You need to design and implement an integration that will handle these CSV files efficiently and ensure they are correctly processed by **System B**.

Here are the requirements:

3. The integration should pick up CSV files from a specific directory on a secure FTP server.
4. After processing, the files should be archived in a different directory for auditing purposes.
5. If a file fails to process, it should be moved to an error directory for troubleshooting.
6. The integration should send a notification email to the Operations team upon successful or failed processing of the files.

Design and implement an integration that will handle these CSV files efficiently and ensure they are correctly processed.

Note: System A: Oracle CX Sales Cloud, System B: Oracle RightNow.

Use the FTP Adapter to read the files from the secure FTP server directory to trigger the integration, process them by using Stage File Action operations to System B, and use the FTP Adapter to move the processed file to the archive directory. Configure error handling to move failed files to the error directory. Use the Notification action to send emails to the Operations

1. **FTP Adapter Configuration:**
    - Set up the FTP Adapter to monitor the secure FTP server directory for new CSV files.
    - Use the FTP Adapter to read the incoming files and trigger the integration.
2. **Process the Files:**
    - For each file, define the necessary actions to process it (e.g., parsing the CSV, transforming data, and invoking System B).
3. **Move Files to Archive/Error Directories:**
    - After processing a file, use the FTP Adapter to move the file to an archive directory if successful.
    - If a file fails to process, move it to an error directory for further investigation.
4. **Error Handling:**
    - Configure error handling to ensure that failed files are detected and moved to the error directory automatically.
    - Implement retries or logging mechanisms to ensure smooth recovery of failed files.
5. **Notification:**
    - Add Notification actions to send email alerts to the Operations team for both successful and failed file processing.