Sure, I can help with that! Oracle Cloud HCM (Human Capital Management) is a comprehensive suite of applications designed to manage human resources processes. Here's a brief overview of how positions, departments, and jobs are structured within Oracle Cloud HCM:

**Positions**

* **Positions** represent specific roles within an organization. Each position can be associated with a job and a department.
* Positions can be full-time, part-time, or temporary.
* They include details like job title, location, and reporting relationships.

**Departments**

* **Departments** are organizational units within a company, such as Marketing, Finance, or Human Resources.
* Each department can have multiple positions and employees.
* Departments help in organizing and managing employees based on their functional areas.

**Jobs**

* **Jobs** define the general roles and responsibilities within the organization.
* A job can be associated with multiple positions.
* Jobs include details like job title, job description, and required qualifications.

**How They Interrelate**

* **Positions** are specific instances of **jobs** within **departments**.
* For example, the job "Software Engineer" might have multiple positions in the "IT Department," each with different responsibilities or locations.

Lookups in Oracle Cloud are a way to define and manage lists of values that users can select from when entering data. They help standardize data entry and ensure consistency across the application. Here’s a more detailed look at lookups in Oracle Cloud:

**What Are Lookups?**

* **Lookups** are predefined lists of values that users can choose from in various fields within Oracle Cloud applications.
* They are used to ensure data consistency and accuracy by providing a controlled set of options.

**Types of Lookups**

* **Standard Lookups**: These are predefined by Oracle and are used across different modules.
* **Common Lookups**: These are shared across multiple applications within Oracle Cloud.
* **User-Defined Lookups**: These are custom lookups created by users to meet specific business needs.

**Components of Lookups**

* **Lookup Type**: This defines the category or group of the lookup values (e.g., Country Codes, Job Titles).
* **Lookup Code**: This is the actual value that users select (e.g., US for United States, ENG for Engineer).
* **Meaning**: This provides a description or label for the lookup code (e.g., United States, Engineer).
* **Enabled Flag**: This indicates whether the lookup value is active and available for use.

**Managing Lookups**

* Lookups can be managed through the Oracle Cloud interface, where administrators can add, modify, or disable lookup values as needed.
* They can also set up validation rules to ensure that only valid lookup values are used.

**Use Cases**

* **Data Entry**: Ensuring users select from a predefined list of values, reducing errors.
* **Reporting**: Standardized data makes it easier to generate accurate reports.

**There are three different types of common lookups available in Oracle Fusion that decide the customization for the lookup:**

* 1. User lookup – This is a flexible type of lookup; you can add new values and enable/disable the existing values to it.
* 2. Extensible lookup – In this type of lookup, you can add new values but cannot disable the existing values.
* 3. System lookup – System lookup does not allow you to add or disable the existing values.

In Oracle Cloud, base tables and translated tables are used to store data in a structured and efficient manner. Here's an overview of each:

**Base Table**

* **Definition**: A base table is the primary table that stores the core data for a particular entity or module within Oracle Cloud.
* **Example**: For an employee entity, the base table might include columns for employee ID, name, job title, department, and other essential attributes.

**Translated Table**

* **Definition**: A translated table is used to store language-specific translations of data that is stored in the base table.
* **Example**: For the same employee entity, the translated table might include columns for employee ID (as a reference to the base table), language code, and translated values for attributes like job title or department name.

**Example Scenario**

* **Base Table**: Stores the job title "Software Engineer" with an ID of 101.
* **Translated Table**: Stores translations for job title ID 101, such as "Ingeniero de Software" for Spanish and "Ingénieur Logiciel" for French.

**GRADES**

In Oracle Cloud, grades are used to define the levels or ranks within an organization. They help in categorizing employees based on their roles, responsibilities, and compensation levels. Here’s a detailed look at grades in Oracle Cloud:

**Grades**

* **Definition**: Grades represent the hierarchical levels within an organization, often associated with specific job roles and compensation ranges.
* **Purpose**: They help in standardizing job levels, facilitating compensation management, and ensuring consistency in employee classification.

**Key Components**

* **Grade Name**: The name or title of the grade (e.g., Grade A, Grade 1).
* **Grade Code**: A unique identifier for the grade.
* **Grade Description**: A brief description of the grade, outlining its purpose and scope.
* **Grade Rate**: The compensation range associated with the grade, which can include minimum, midpoint, and maximum salary values.
* **Grade Steps**: Optional predefined steps within a grade that represent incremental levels of compensation or seniority.

**How Grades Are Used**

* **Job Assignments**: Employees are assigned to grades based on their job roles and responsibilities.
* **Compensation Management**: Grades help in managing salary structures, ensuring that employees are compensated fairly based on their grade level.
* **Career Progression**: Grades provide a clear framework for career advancement, allowing employees to understand the potential growth paths within the organization.

**Example Scenario**

* **Grade A**: Entry-level positions with a salary range of \$30,000 to \$40,000.
* **Grade B**: Mid-level positions with a salary range of \$40,000 to \$60,000.
* **Grade C**: Senior-level positions with a salary range of \$60,000 to \$80,000.
* Find Duplicate records in a table: select, count, group by, having
* Set operators – union(then what is the purpose of union)
* How the ddl dml operations used
* Between
* Pl/sql
* Advantages of cursor , predefined cursor, how do we pass parameters to cursor
* Trigger , example
* Main diff bw sql and plsql
* Why do I write a procedure
* Diff bw procedure and functions
* What is package , how do I write a package
* All are public procedures that are in package
* Exception types. When I need to have predefined , user exceptions
* When we don’t know about which exception,then we can use when others exception . it will catch that exception
* Diff types of persons(employee,contingent, non ) and what are the assignments for it (e,c)
* Tables (per\_persons\_f, per all assignments etc…)
* Can you tell if I want to create a custom report . what is the navigation?(and see how to assign the parameters also) how u will attach lov
* How do you download xml
* What extension it will be
* After saving how u will create a template
* What outputs I can make
* How bi publisher works
* Templates work?
* Data models work?
* How you load data?

In PL/SQL, a **cursor** is a pointer that allows you to fetch and process rows returned by a SQL query. There are two main types of cursors: **implicit cursors** and **explicit cursors**.

**Implicit Cursors**

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed. You don't have direct control over these cursors, but you can use certain attributes to get information about the execution.

**Attributes**:

* **%FOUND**: Returns TRUE if the SQL statement affected one or more rows.
* **%NOTFOUND**: Returns TRUE if the SQL statement affected no rows.
* **%ISOPEN**: Always returns FALSE for implicit cursors because they are automatically closed after execution.
* **%ROWCOUNT**: Returns the number of rows affected by the SQL statement[[1]](https://www.tutorialspoint.com/plsql/plsql_cursors.htm).

**Explicit Cursors**

Explicit cursors are defined by the programmer and provide more control over the execution cycle. You can declare, open, fetch, and close explicit cursors.

**Steps to Use Explicit Cursors**:

1. **Declare**: Define the cursor in the declaration section.

CURSOR cursor\_name IS

SELECT column1, column2 FROM table\_name WHERE condition;

1. **Open**: Open the cursor to execute the query.

OPEN cursor\_name;

1. **Fetch**: Retrieve rows from the cursor into variables.

FETCH cursor\_name INTO variable1, variable2;

1. **Close**: Close the cursor to release resources.

CLOSE cursor\_name;

**Example**:

DECLARE

CURSOR emp\_cursor IS

SELECT employee\_id, first\_name, last\_name FROM employees WHERE department\_id = 10;

emp\_id employees.employee\_id%TYPE;

emp\_first employees.first\_name%TYPE;

emp\_last employees.last\_name%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_id, emp\_first, emp\_last;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(emp\_id || ' ' || emp\_first || ' ' || emp\_last);

END LOOP;

CLOSE emp\_cursor;

END;

This example demonstrates how to declare, open, fetch, and close an explicit cursor to retrieve employee details from a specific department[[2]](https://www.oracletutorial.com/plsql-tutorial/plsql-cursor/)[[1]](https://www.tutorialspoint.com/plsql/plsql_cursors.htm).

Would you like to dive deeper into any specific aspect of cursors in PL/SQL?

**References**

[1] [PL/SQL Cursors - Online Tutorials Library](https://www.tutorialspoint.com/plsql/plsql_cursors.htm)

[2] [PL/SQL Cursor By Practical Examples - Oracle Tutorial](https://www.oracletutorial.com/plsql-tutorial/plsql-cursor/)

**Position synchronization** in Human Capital Management (HCM) refers to the process where certain attributes of an employee's assignment are automatically inherited from the associated position. This helps maintain consistency and accuracy in employee data.

Tables

When an employee is terminated in Oracle HCM, several tables can be affected to reflect the change in their employment status. Here are some key tables that are typically impacted:

1. **PER*ALL*ASSIGNMENTS\_F**: This table stores information about employee assignments. The ASSIGNMENT\_STATUS\_TYPE column will be updated to reflect the termination status [[1]](https://community.oracle.com/customerconnect/discussion/585548/need-sql-query-table-names-to-get-termination-and-reverse-termination-details-of-an-employee).
2. **PER*ALL*PEOPLE\_F**: This table contains core personal data for individuals. The EFFECTIVE\_END\_DATE column will be updated to indicate the end of the person's employment[[1]](https://community.oracle.com/customerconnect/discussion/585548/need-sql-query-table-names-to-get-termination-and-reverse-termination-details-of-an-employee).
3. **PER*PERIODS*OF\_SERVICE**: This table tracks the periods of service for employees. A new record may be inserted or an existing record updated to reflect the termination[[1]](https://community.oracle.com/customerconnect/discussion/585548/need-sql-query-table-names-to-get-termination-and-reverse-termination-details-of-an-employee).
4. **PER*WORK*RELATIONSHIPS**: This table stores information about the work relationships of employees. The STATUS column will be updated to reflect the termination[[2]](https://community.oracle.com/customerconnect/discussion/837422/query-to-fetch-work-relationship-data-for-terminated-employee-in-oracle-fusion-hcm).
5. **PER*TERMINATIONS*F**: This table specifically records termination details, including the termination date, reason, and other relevant information[[2]](https://community.oracle.com/customerconnect/discussion/837422/query-to-fetch-work-relationship-data-for-terminated-employee-in-oracle-fusion-hcm).

**Example Query to Check Termination Status:**

Here's an example query to check the termination status of an employee:

SELECT

papf.PERSON\_ID,

papf.PERSON\_NUMBER,

papf.EFFECTIVE\_END\_DATE,

paf.ASSIGNMENT\_STATUS\_TYPE,

pts.TERMINATION\_DATE,

pts.TERMINATION\_REASON

FROM

PER\_ALL\_PEOPLE\_F papf

JOIN

PER\_ALL\_ASSIGNMENTS\_F paf ON papf.PERSON\_ID = paf.PERSON\_ID

LEFT JOIN

PER\_TERMINATIONS\_F pts ON papf.PERSON\_ID = pts.PERSON\_ID

WHERE

papf.PERSON\_ID = :employee\_id

AND

papf.EFFECTIVE\_END\_DATE IS NOT NULL;

This query retrieves the termination status, assignment status, and termination details for a specific employee.

Would you like more details on how to use these tables or specific examples of queries? [[1]](https://community.oracle.com/customerconnect/discussion/585548/need-sql-query-table-names-to-get-termination-and-reverse-termination-details-of-an-employee): Oracle Documentation [[2]](https://community.oracle.com/customerconnect/discussion/837422/query-to-fetch-work-relationship-data-for-terminated-employee-in-oracle-fusion-hcm): Oracle Documentation

**References**

[1] [Need SQL Query/ table names to get termination and reverse termination ...](https://community.oracle.com/customerconnect/discussion/585548/need-sql-query-table-names-to-get-termination-and-reverse-termination-details-of-an-employee)

[2] [query to fetch work relationship data for terminated employee in oracle ...](https://community.oracle.com/customerconnect/discussion/837422/query-to-fetch-work-relationship-data-for-terminated-employee-in-oracle-fusion-hcm)

To get the manager name of an employee in Oracle HCM, you typically need to join several tables. Here are the key tables involved:

1. **PER*ALL*ASSIGNMENTS\_F**: This table contains assignment details, including the SUPERVISOR\_ID which links to the manager's PERSON\_ID.
2. **PER*PERSON*NAMES\_F**: This table holds the names of individuals, which you can use to retrieve the manager's name.
3. **PER*ALL*PEOPLE\_F**: This table contains core personal data for individuals, which can be used to get additional details about the manager.

**Example Query:**

Here's an example of how you can join these tables to get the manager's name for each employee:

SELECT

emp.PERSON\_ID AS Employee\_ID,

emp.FIRST\_NAME AS Employee\_First\_Name,

emp.LAST\_NAME AS Employee\_Last\_Name,

mgr.PERSON\_ID AS Manager\_ID,

mgr.FIRST\_NAME AS Manager\_First\_Name,

mgr.LAST\_NAME AS Manager\_Last\_Name

FROM

PER\_ALL\_ASSIGNMENTS\_F emp\_assign

JOIN PER\_PERSON\_NAMES\_F emp ON emp\_assign.PERSON\_ID = emp.PERSON\_ID

JOIN PER\_ALL\_ASSIGNMENTS\_F mgr\_assign ON emp\_assign.SUPERVISOR\_ID = mgr\_assign.PERSON\_ID

JOIN PER\_PERSON\_NAMES\_F mgr ON mgr\_assign.PERSON\_ID = mgr.PERSON\_ID

WHERE emp.EFFECTIVE\_END\_DATE IS NULL

AND mgr.EFFECTIVE\_END\_DATE IS NULL

AND emp\_assign.EFFECTIVE\_END\_DATE IS NULL

AND mgr\_assign.EFFECTIVE\_END\_DATE IS NULL;

**Explanation:**

* **SELECT**: Specifies the columns to retrieve, including employee and manager IDs and names.
* **FROM PER*ALL*ASSIGNMENTS*F emp*assign**: Indicates the primary table for employee assignments and assigns it an alias (emp\_assign).
* **JOIN PER*PERSON*NAMES*F emp ON emp*assign.PERSON*ID = emp.PERSON*ID**: Joins the employee names table to get the employee's name.
* **JOIN PER*ALL*ASSIGNMENTS*F mgr*assign ON emp*assign.SUPERVISOR*ID = mgr*assign.PERSON*ID**: Joins the assignments table again to get the manager's assignment details.
* **JOIN PER*PERSON*NAMES*F mgr ON mgr*assign.PERSON*ID = mgr.PERSON*ID**: Joins the manager names table to get the manager's name.
* **WHERE**: Filters the results to include only current records.

This query retrieves the names and IDs of employees along with their managers.

**Bursting** in HCM (Human Capital Management) refers to the process of splitting data into smaller, manageable chunks and then generating and delivering documents for each chunk. This is particularly useful for generating personalized reports or notifications for employees based on specific criteria.

**Key Steps in Implementing Bursting in HCM:**

1. **Define the Data Model**:
   * Create or modify the data model to include the necessary data sets.
   * Ensure the data set is sorted or grouped by the element you want to use for splitting the data (e.g., EMPLOYEE\_ID).
2. **Add Bursting Definition**:
   * **Split By**: Define the element that will govern how the data is split. For example, to split by employee, you might use EMPLOYEE\_ID.
   * **Deliver By**: Define the element that will govern how the documents are formatted and delivered. This could also be EMPLOYEE\_ID if each employee receives a personalized document[[1]](https://docs.oracle.com/cd/E21764_01/bi.1111/e18862/T527073T555155.htm).
3. **Configure Delivery Options**:
   * Set up the delivery options such as email addresses, subject lines, and attachment formats.
   * You can configure these options dynamically using a SQL query or statically by hardcoding them in the data model[[1]](https://docs.oracle.com/cd/E21764_01/bi.1111/e18862/T527073T555155.htm).
4. **Modify the Data Model**:
   * Update the data model to include the bursting configuration.
   * Ensure the Split By and Deliver By fields are correctly set up in the bursting section of the data model[[2]](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/).
5. **Reference the Data Model in BI Report**:
   * Reference the modified data model in the BI report to ensure the bursting setup is applied.
   * Test the bursting functionality by running the report and verifying that documents are generated and delivered as expected[[2]](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/).

**Example Use Cases:**

* **Employee Notifications**: Send personalized notifications to employees or managers upon events like new hires, terminations, or performance document completions.
* **Payslip Generation**: Generate and email payslips to employees based on a single data extract[[2]](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/).

Would you like more detailed steps on setting up bursting for a specific report or extract in Oracle HCM?

[[1]](https://docs.oracle.com/cd/E21764_01/bi.1111/e18862/T527073T555155.htm): [Oracle Documentation](https://docs.oracle.com/cd/E21764_01/bi.1111/e18862/T527073T555155.htm) [[2]](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/): [Fusion HCM Knowledgebase](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/)

**References**

[1] [Adding Bursting Definitions - Oracle](https://docs.oracle.com/cd/E21764_01/bi.1111/e18862/T527073T555155.htm)

[2] [Implementing Bursting feature for HCM Extract - Fusion HCM Knowledgebase](https://fusionhcmknowledgebase.com/2020/10/implementing-bursting-for-hcm-extract/)

 a **report** is a tool used to present data in a structured and interactive format, allowing users to analyze and gain insights from their transactional data