JPQL (Jakarta Persistence Query Language) es un lenguaje de consulta utilizado en el entorno de Jakarta EE (anteriormente Java EE) para interactuar con bases de datos relacionales de una manera orientada a objetos. A diferencia de SQL, que opera directamente sobre tablas y columnas de la base de datos, JPQL opera sobre entidades y sus relaciones, tal como están definidas en tu modelo de objetos Java. Esto significa que, en lugar de escribir consultas para seleccionar datos de una tabla clientes, escribes consultas para seleccionar instancias de tu clase Cliente.

**Características clave de JPQL:**

* **Orientado a objetos**: JPQL permite escribir consultas utilizando los nombres de tus clases de entidad y sus atributos, en lugar de los nombres de tablas y columnas de la base de datos subyacente. Esto proporciona una capa de abstracción que hace que las consultas sean más intuitivas para los desarrolladores Java y las independiza de los detalles específicos del esquema de la base de datos.
* **Independencia de la base de datos**: Dado que JPQL opera sobre el modelo de objetos, el proveedor de persistencia (como Hibernate o EclipseLink) es responsable de traducir las consultas JPQL a SQL nativo para la base de datos subyacente. Esto significa que puedes cambiar de base de datos sin tener que reescribir tus consultas.
* **Sintaxis similar a SQL**: La sintaxis de JPQL es muy similar a la de SQL, lo que facilita su aprendizaje para los desarrolladores familiarizados con SQL. Incluye cláusulas como SELECT, FROM, WHERE, ORDER BY, GROUP BY, y HAVING.
* **Manejo de relaciones**: JPQL permite navegar por las relaciones entre entidades. Por ejemplo, si una entidad Pedido tiene una relación con una entidad Cliente, puedes escribir una consulta JPQL que seleccione pedidos realizados por un cliente específico o incluso acceder a los atributos del cliente directamente desde la consulta de pedido.
* **Parámetros de consulta**: JPQL soporta el uso de parámetros en las consultas, lo que ayuda a prevenir ataques de inyección SQL y mejora la legibilidad. Estos parámetros pueden ser posicionales (ej. ?1) o con nombre (ej. :nombreParametro).
* **Consultas dinámicas y estáticas**:
  + **Consultas estáticas (Named Queries)**: Se definen en la clase de entidad usando anotaciones (@NamedQuery o @NamedNativeQuery) o en un archivo XML. Son útiles para consultas que se utilizan repetidamente, ya que se analizan una sola vez en el momento de la carga de la aplicación, lo que puede mejorar el rendimiento.
  + **Consultas dinámicas**: Se construyen y ejecutan en tiempo de ejecución utilizando el método createQuery() del EntityManager. Son más flexibles para situaciones donde la consulta necesita ser construida dinámicamente basándose en la lógica de la aplicación.
* **Retorno de resultados**: Las consultas JPQL pueden devolver entidades completas, partes de entidades (proyecciones de campos específicos) o incluso resultados de tipo Object[] o Tuple cuando se seleccionan múltiples tipos de datos. TypedQuery<T> se utiliza para obtener resultados de un tipo específico, proporcionando verificación de tipo en tiempo de compilación.

En resumen, JPQL es una herramienta fundamental en JPA que simplifica el acceso y la manipulación de datos en aplicaciones Java, permitiendo a los desarrolladores trabajar con objetos en lugar de con las complejidades de la base de datos subyacente.