?What is Hibernate?

Ans. Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration files.

?Hibernate VS JDBC?

Ans. **Hibernate→**is database independent, no need to take care of mapping tables to database as hbm.XML file will do it.

JDBC→is database dependent, need to create tables in DB.

Hibernate does not require an application server to operate.

?What is DOCTYPE?

Ans. DOCTYPE →a document type declaration is an instruction that associates a particular XML file with a document type definition.

?Explain hbm.xml in detail?

Ans.
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
```

```
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<classname="Employee"table="EMPLOYEE">
<metaattribute="class-description">
      This class contains the employee detail.
</meta>
<idname="id"type="int"column="id">
<generatorclass="native"/>
</id>
<propertyname="firstName"column="first_name"type="string"/>
<propertyname="lastName"column="last_name"type="string"/>
<propertyname="salary"column="salary"type="int"/>
</class>
</hibernate-mapping>
```

You should save the mapping document in a file with the format <classname>.hbm.xml. We saved our mapping document in the file Employee.hbm.xml. Let us see little detail about the mapping elements used in the mapping file:

- The mapping document is an XML document having **<hibernate-mapping>** as the root element which contains all the <class> elements.
- The **<class>** elements are used to define specific mappings from a Java classes to the database tables. The Java class name is specified using the **name** attribute of the class element and the database table name is specified using the **table** attribute.
- The **<meta>** element is optional element and can be used to create the class description.
- The **<id>** element maps the unique ID attribute in class to the primary key of the database table. The **name** attribute of the id element refers to the property in the class and the **column** attribute refers to the column in the database table. The **type** attribute holds the hibernate mapping type, this mapping types will convert from Java to SQL data type.
- The **<generator>** element within the id element is used to automatically generate the primary key values. Set the **class** attribute of the generator element is set to **native** to let hibernate pick up either **identity, sequence** or **hilo** algorithm to create primary key depending upon the capabilities of the underlying database.
- The **<property>** element is used to map a Java class property to a column in the database table. The **name** attribute of the element refers to the property in the class and the **column** attribute refers to the column in the database table. The **type** attribute holds the hibernate mapping type, this mapping types will convert from Java to SQL data type.

?What is Dialect?

Ans. Dialect is an interpreter which converts HQL Query to the native query. (It depends which dialect you are using). It converts Object to SQL and SQL to Object.

?What is Cascading?

Ans. Cascading removed Child object when Parent Object is removed. In Hibernate we define it in hbm.xml in <Set> tag.

?What is **show_sql** property doing in cfg.xml?

Ans. It logs all the generated SQL statements to the console.

hibernate.c3p0.min_size – Minimum number of JDBC connections in the pool. Hibernate default: 1

hibernate.c3p0.max_size – Maximum number of JDBC connections in the pool. Hibernate default: 100

hibernate.c3p0.timeout – When an idle connection is removed from the pool (in second). Hibernate default: 0, never expire.

?What is the use of Hibernate.cfg.xml?

Ans. Is a configuration file that contain properties required to establish connection with database.

?What is <mapping resource > ?

Ans. Contains the path of hbm.xml mapping

?what is hbm.xml?

Ans. It is mapping file which instructs hibernate to map defined class to database.

First level cache is associated with "session" object. The scope of cache objects is of session. Once session is closed, cach ed objects are gone forever. First level cache is enabled by default and you can not disable it. When we query an entity first time, it is retrieved from database and stored in first level cache associated with hibernate session. If we query same object again with same session object, it will be loaded from cach e and no sql query will be executed. The loaded entity can be removed from session using evict() method. The next loading of this entity will again make a database call if it has b een removed using evict() method. The whole session cache can be removed using clear() method. It will remove all the entities stored in cache.(TO REDUCE THE NUMBER OF SQL QUERIES)

Second level cache is apart from first level cache which is available to be used globally in session factory scope. second le vel cache is created in session factory scope and is available to be used in all sessions which are created using that particular session factory. It also me ans that once session factory is closed, all cache associated with it die and cache manager also closed down. Whenever hiber nate session try to load an entity, the very first place it look for cached copy of entity in first level cache (associated with particular hibernate session). If cached copy of entity is presen t in first level cache, it is returned

as result of load method. If there is no cached entity in first level cache, then second level cache is looked up for cached entity. If second level cache has cached entity, it is returned as result of load method. But, before returning the entity, it is stored in first level ca che also so that next invocation to load method for entity will return the entity from first level cache itself, and there will not be need to go to second level cach e again. If entity is not found in first level cache and second level cache also, then database query is executed and entity is stored in both cache levels, before returning as response of load() method.(TO REDUCE DATABASE TRAFFIC.)

?How many types of Inheritance mapping are there?

Ans. 1.> Table Per Hierarchy→Single table is required to map the whole hierarchy and discriminator column is added to identify the class.

2→ Table Per Concrete Class-> In this tables created will have no relation with each other.

3→Table Per Subclass→ Subclass mapped table are related to parent class table by primary key and foreign key relation ship.

?What is model factory?

Ans. Model Factory is a factory of service classes(also called DAO factory). It follows Factory Design Pattern i.e if there are multiple implementation then it provides access to the user that which implementation class to instantiate.

?What is Transaction?

Ans. Transaction is an interface→

- Allows the application to define unit of work.
- Is a set of Database calls

? Transaction transaction = **session.beginTransaction?**

Ans. Begin a unit of work and return associated transaction object.

#Session.save→ Persist the given transient instance , first assigning a generated identifier.

#Session.savevsSession.persist→ Both make the transient instance persist but "Save" return serializable object where as "Persist" return void.

#Session.updatevsSession.merge→At first look both update() and merge() methods seems similar because both of them are used to convert the object which is in detached state into persistence state, but the major difference between update and merge is that update method cannot be used when the same object exists in the session.

Use "Update" if you are sure that session does not contain persistant object with the same identifier. In case of merge we need not worry about state of session.

?In which case Rollback occurs?

- Loss of communication with participant in case of distributed transactions
- An invalid or out of range parameter

?What is Criteria?

Ans. It is an interface for retrieving entities by composing criterian objects. It is used for performing “search” operations.

```
 List cats = session.createCriteria(Cat.class)
     .add(Restrictions.like("name", "Iz%") )
     .add( Restrictions.gt( "weight", new Float(minWeight) ) )
     .addOrder(Order.asc("age") )
     .list();
```

?Criteriavs HQL?

- HQL is to perform both select and non-select operations on the data, but Criteria is only for selecting the data, we cannot perform non-select operations using criteria.
- HQL is suitable for executing Static Queries, where as Criteria is suitable for executing Dynamic Queries
- HQL doesn’t support pagination concept, but we can achieve pagination with Criteria.
- Criteria used to take more time to execute than HQL.

The Hibernate Session interface provides createCriteria() method which can be used to create a Criteria object that returns instances of the persistence object's class when your application executes a criteria query.

?Session.get() vsSession.load()

Ans. Both are used for retrieving object from database. But in case of get() we get original record and if no record is found than it will return NULL.

In case of Load() we will get proxy object and in case if no record is found then we will get "ObjectNotFound" exception.

Proxy means, hibernate will prepare some fake object with given identifier value in the memory without hitting the database

?What is Session Factory?

Ans. It is a DCP which is created by config settings in hibernate.cfg.xml. It is immutable.

?What is Session?

Ans. Session is an interface. The main runtime interface between a Java application and Hibernate. A Session is used to get a physical connection with a database.

It all depends on how you obtain the session→

if you use sessionFactory.getCurrentSession(), you'll obtain a "current session" which is bound to the lifecycle of the transaction and will be automatically flushed and closed when the transaction ends (commit or rollback).

if you decide to use sessionFactory.openSession(), you'll have to manage the session yourself and to flush and close it "manually".

?What is Dirty checking in Hibernate?

Ans. The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction.

Let's understand it by the example given below:

SessionFactory factory = cfg.buildSessionFactory();

```
Session session1 = factory.openSession();

Transaction tx=session2.beginTransaction();

Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));

e1.setSalary(70000);

tx.commit();

session1.close();
```

Here, after getting employee instance e1 and we are changing the state of e1.

After changing the state, we are committing the transaction. In such case, state will be updated automatically. This is known as dirty checking in hibernate.

?What is difference between attached and detached criteria in hibernate?

Ans. The detached criteria allows you to create the query without Session. Using a DetachedCriteria is exactly the same as a Criteria except you can do the initial creation and setup of your query without having access to the session. When it comes time to run your query, you must convert it to an executable query with getExecutableCriteria(session).

?Dynamic-insert & Dynamic-update?

Ans.

 `dynamic-update` (defaults to `false`): Specifies that `UPDATE` SQL should be generated at runtime and contain only those columns whose values have changed
•`dynamic-insert` (defaults to `false`): Specifies that `INSERT` SQL should be generated at runtime and contain only the columns whose values are not null.

?What do you mean by Named – SQL query?

Named SQL queries are defined in the mapping xml document and called wherever required.

Example:

```
<sql-query name = "empdetails">
.E   <return alias="emp" class="com.testmployee"/>
     SELECT emp.EMP_ID AS {emp.empid},
emp.EMP_ADDRESS AS {emp.address},
emp.EMP_NAME AS {emp.name}
FROM Employee EMP WHERE emp.NAME LIKE :name
</sql-query>
```

Invoke Named Query :

```
List people = session.getNamedQuery("empdetails")
.setString("TomBrady", name)
.setMaxResults(50)
.list();
```

?If you want to see the Hibernate generated SQL statements on console, what should we do?

In Hibernate configuration file set as follows:

```
<property name="show_sql">true</property>
```

identity

This is database dependent, actually its not working in oracle

In this case (identity generator) the id value is generated by the database, but not by the hibernate, but in case of increment hibernate will take over this

this identity generator doesn’t needs any parameters to pass

this identity generator is similar to increment generator, but the difference was increment generator is database independent and hibernate uses a select operation for selecting max of id before inserting new record

But in case of identity, no select operation will be generated in order to insert an id value for new record by the hibernate

?Why Caching?

Ans. Application Performance Optimization. A cache is designed to reduce the traffic between our application and database.

?session.evict(0)→ will remove object of 0th Position from session.

#For Second Level cache→

- Add 2 configuration setting in hibernate.cfg.xml file

<property name="cache.provider_class">org.hibernate.cache.**EhCacheProvider**</property>

<property name="hibernate.cache.use_second_level_cache">true</property>

- Add cache usage setting in hbm file

```
<cache usage="read-only" />
```

(means the current session has all the permissions on object but other session can only read it.

?How to enable second-level cache?

Ans. There are four ways to use second level cache. <cache usage="read-only" />

None:

read-only: caching will work for read only operation.

nonstrict-read-write: caching will work for read and write but one at a time.

read-write: caching will work for read and write, can be used simultaneously.

transactional: caching will work for transaction.

- #Query-Cache→
- We use query cache in case if query is executed again and again because it holds query along with its result set. Cache contains original result rather than persistent object.
- Second-level cache should be enabled.
- Set the hibernate.cache.use_query_cache property to true.
- Need to set permissions using →Query.setCacheable(true);

(means for this “Query” Query-cache will work)

One of greatest dis-advantage of using hibernate is that if the value is stored in cache and that particular value is if changed by someone in DB then while trying to get value it will return value stored in cache and it will not look in DB.

?. What is detached object?

Ans. Is the one which was associated with session once and its session is not alive now.

?How can you update detached oobject?

Ans. To update detachedobj we will open another session and we will call session.update method.

?What is HQL?

Ans. Is Hibernate Query Language. It is database independent means queries written in HQL can be used for any database.

?.What are Callback interfaces?

Callback interfaces allow the application to receive a notification when something interesting happens to an object—for example, when an object is loaded, saved, or deleted. Hibernate applications don't need to implement these callbacks, but they're useful for implementing certain kinds of generic functionality.

?What is Lazy Fetching?

Ans. Lazy=”true”. Means it will bring only recommended data or data on demand.

If Lazy=”false”. It will bring all the data. Even data from FK tables. This will increase memory consumption.

?What is Eager Fetching?

Ans. Fetch=”join”. Single join query will bring all the data.

?What is Batch Fetching?

Ans. Batch Fetching will fetch data with the help of PK and FK. It is enabled by-default.

?What do you mean by fetching Strategy?

Ans. Fetching Strategy declare how many queries will run for relation.

?.How can a whole class be mapped as immutable?

Ans. Mark the class as mutable="false" (Default is true),. This specifies that instances of the class are (not) mutable. Immutable classes, may not be updated or deleted by the application.

?How can Hibernate be configured to access an instance variable directly and not through a setter method ?

Ans. By mapping the property with access="field" in Hibernate metadata. This forces hibernate to bypass the setter method and access the instance variable directly while initializing a newly loaded object.

?inverse=”true”?

Ans. it defines which side is the parent or the relationship owner for the two entities (parent or child). Hence, inverse="true" in a Hibernate mapping shows that this class (the one with this XML definition) is the relationship owner; while the other class is the child.

?.What are the ways to express joins in HQL?

Ans. HQL provides four ways of expressing (inner and outer) joins:-

- An implicit association join
- An ordinary join in the FROM clause
- A fetch join in the FROM clause.
- A theta-style join in the WHERE clause.
-

?.What are the most common methods of Hibernate configuration?

The most common methods of Hibernate configuration are:

- Programmatic configuration
- XML configuration (hibernate.cfg.xml)

?What does ORM consists of ?

Ans. An ORM solution consists of the followig four pieces->

- API for performing basic CRUD operations
- API to express riesrefering to classes
- Facilities to specify metadata
- Optimization facilities : dirty checking,lazy associations fetching

?What is the difference between criteria and detached criteria?

Ans. Criteria is attached with session at compile time were as detached criteria is attached (can be) with session at run time.

?What is the role of Session Interface?

Ans.

- Wraps a JDBC connection
- Factory of transaction
- Holds First level- cache

?What is the use of Stored Procedure?

Ans. Stored procedure is used for Time-Critical Reports.

?How to call stored procedure in Hibernate?

Ans.

- Using JDBC connection
- Using native SQL query
- Using named query

?What is candidate key?

Ans. A not null key which a candidate for Primary key.

?What is composite key?

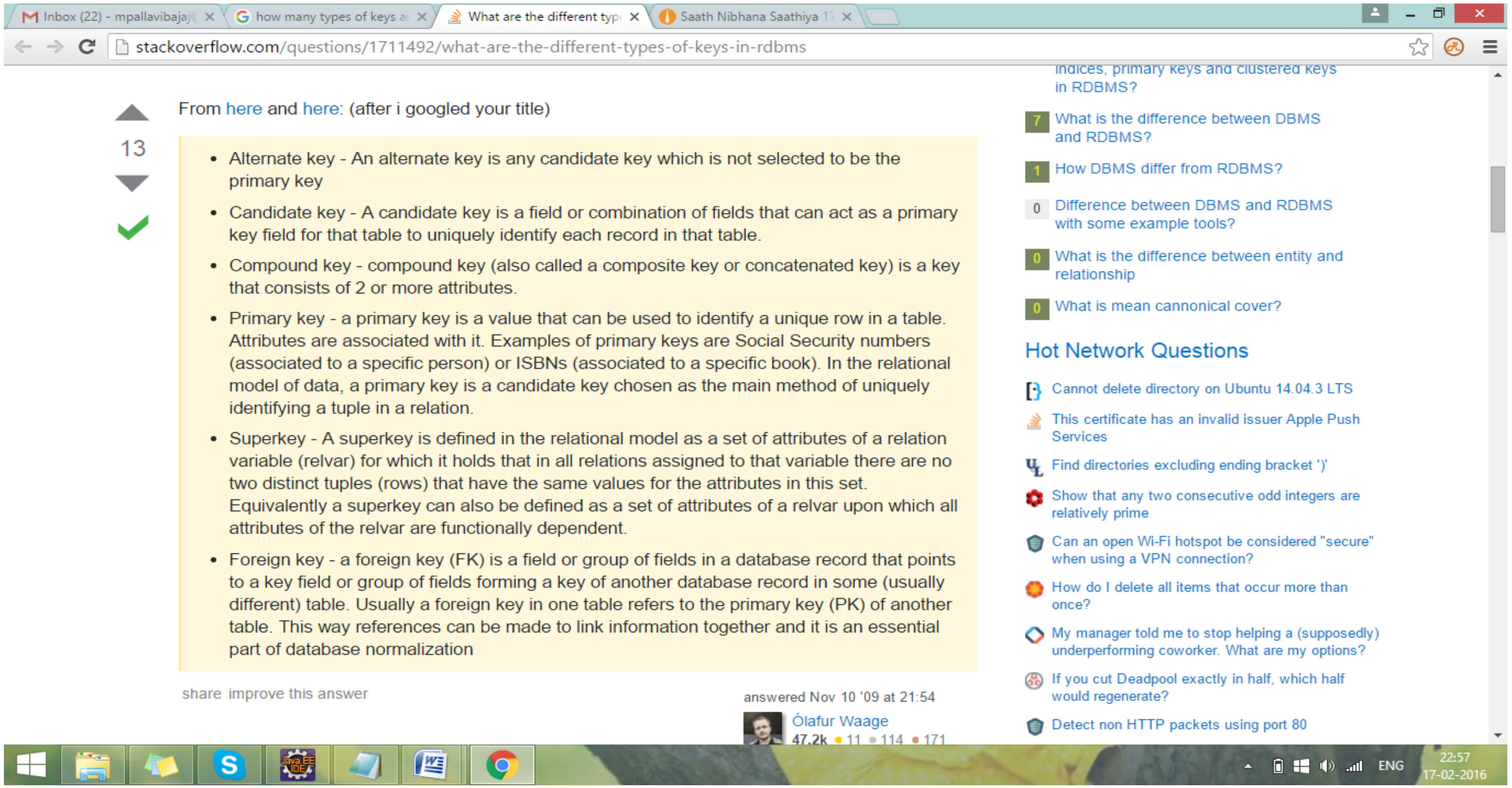
Ans. A Primary which consist of two or more attributes is called composite key. We define it using :

<composite-id>

I have never used it because I am using non-business primary key.

Super Key

A Super key is any combination of fields within a table that uniquely identifies each record within that table.


Inbox (22) - mpallavibaja
how many types of keys a
What are the different typ
Saath Nibhana Saathiya 1
stackoverflow.com/questions/1711492/what-are-the-different-types-of-keys-in-rdbms
13

From here and here: (after i googled your title)

- Alternate key - An alternate key is any candidate key which is not selected to be the primary key
- Candidate key - A candidate key is a field or combination of fields that can act as a primary key field for that table to uniquely identify each record in that table.
- Compound key - compound key (also called a composite key or concatenated key) is a key that consists of 2 or more attributes.
- Primary key - a primary key is a value that can be used to identify a unique row in a table. Attributes are associated with it. Examples of primary keys are Social Security numbers (associated to a specific person) or ISBNs (associated to a specific book). In the relational model of data, a primary key is a candidate key chosen as the main method of uniquely identifying a tuple in a relation.
- Superkey - A superkey is defined in the relational model as a set of attributes of a relation variable (relvar) for which it holds that in all relations assigned to that variable there are no two distinct tuples (rows) that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relvar upon which all attributes of the relvar are functionally dependent.
- Foreign key - a foreign key (FK) is a field or group of fields in a database record that points to a key field or group of fields forming a key of another database record in some (usually different) table. Usually a foreign key in one table refers to the primary key (PK) of another table. This way references can be made to link information together and it is an essential part of database normalization

share improve this answer

answered Nov 10 '09 at 21:54
Ólafur Waage
47.2k ● 11 ● 114 ● 171

indices, primary keys and clustered keys in RDBMS?

7 What is the difference between DBMS and RDBMS?

1 How DBMS differ from RDBMS?

0 Difference between DBMS and RDBMS with some example tools?

0 What is the difference between entity and relationship

0 What is mean cannonical cover?

Hot Network Questions

- Cannot delete directory on Ubuntu 14.04.3 LTS
- This certificate has an invalid issuer Apple Push Services
- Find directories excluding ending bracket ')'
- Show that any two consecutive odd integers are relatively prime
- Can an open Wi-Fi hotspot be considered "secure" when using a VPN connection?
- How do I delete all items that occur more than once?
- My manager told me to stop helping a (supposedly) underperforming coworker. What are my options?
- If you cut Deadpool exactly in half, which half would regenerate?
- Detect non HTTP packets using port 80

Hibernate getCurrentSession

Hibernate `SessionFactory` getCurrentSession() method returns the session bound to the context. But for this to work, we need to configure it in hibernate configuration file like below.

```
<propertyname="hibernate.current_session_context_class">thread</property>
```

Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed. Hibernate Session objects are not thread safe, so we should not use it in multi-threaded environment. We can use it in single threaded environment because it’s relatively faster than opening a new session.

Hibernate openSession

Hibernate `SessionFactory` openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment. For web application frameworks, we can choose to open a new session for each request or for each session based on the requirement.