

### Primary Key

- After @Id can come @GeneratedValue with GenerationType.TABLE|SEQUENCE|IDENTITY|AUTO  
TABLE: there is a own table for the ID generation  
SEQUENCE: an usual DB sequence  
IDENTITY: auto increment (id generated after flush)  
AUTO: persistence provider creates Id
- IdClass (class "only" used for lookups)  
Serializable, no-arg Constructor, hashCode and equals implementation, has no annotations  
In the entity: @IdClass(CustomerPK.class) on table, all fields of the Id with @Id
- EmbeddedId (Entity uses an embedded class)  
like IdClass but with @Embeddable and @Column  
In the entity: reference to the class (not values) and @EmbeddedId on field or getter, @AttributeOverrides can overwrite @Column definitions.

### Inheritance

- Single table per class hierarchy  
@Inheritance(strategy=InheritanceType.SINGLE\_TABLE)  
One big table with a discriminator value (to define, which row was from which class)  
@DiscriminatorValue
- Separate table per subclass  
@Inheritance(strategy=InheritanceType.JOINED)  
The deeper, the more joins are needed, results in slower performance
- Single table per concrete entity class  
Copy columns from parent table. Not supported by EJB 3.0 specification.
- @MappedSuperclass  
The superclass does not have an own table, all columns are inside the child class

### Fetch Type

- @Basic(fetchType=EAGER|LAZY, optional=false)
- optional is for the Schema generator, true = nullable

### EJB QL

- <> for "is not equals"
- JOIN FETCH for eager loading
- JOIN for lazy loading
- @SqlResultSetMapping for native mapping
- Query.setHint("", "") for vendor specific settings
- Subqueries: ALL (all elements fulfill condition), ANY (or SOME, at least one element fulfills condition), EXISTS (there are some)
- UPDATE, INSERT in executeUpdate
- getResultList without SELECT: IllegalArgumentException
- LOWER, UPPER, TRIM, CONCAT, LENGTH, LOCATE, SUBSTRING
- MEMBER OF
- SELECT ... FROM customer AS cust (with or without AS), but DELETE FROM customer cust ... (without AS)

### NamedQuery

- can be used for EJB QL or Native SQL queries
- @NamedQuery (  
name = " findPremiumCustomers",  
query = "SELECT C.ID FROM CUST As C WHERE C.ID >=10")
- ```
<entity-mappings>
  <named-query name="findPremiumCustomers">
    <query>
      SELECT C.ID FROM CUST As C WHERE C.ID >=10
    </query>
  </named-query>
</entity-mappings>
```

### Deployment Descriptor

- ```
<entity class = " Employee">
  <id-class> EmployeePK</id-class>
  <attributes>
    <id name="employeeId"/>
    <id name="employeeName"/>

    <basic name = "address" fetch="LAZY"/>

  </attributes>
</entity>
```