Persistence 2

Viva la persistance!

FIRST8

Wat hebben we geleerd

- Entities
- EntityManager
- Relaties
- Queries
- Transacties

Agenda

Identifiers

Entity Mapping

Locking

Performance

Security

Lunch

Borrel?

Identifiers - Generated

- Waarom?
- Types
  - Table Generation
  - Sequence Generation
  - Identity
  - Auto

Generated Identifiers - Sequence

```
@Entity
@SequenceGenerator(name = "concertGenerator", sequenceName = "SEQ_CONCERT",
                   allocationSize = 10, initialValue = 1)
@Table(name = "CONCERT")
public class Concert implements Serializable {

  @Id
  @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "concertGenerator")
  @Column(name = "ID")
  private Long id;
```

Generated Identifiers - Table

```
@Entity
@TableGenerator(name = "concertGenerator", table= "ID_GEN")
@Table(name = "CONCERT")
public class Concert implements Serializable {

   @Id
   @GeneratedValue(strategy = GenerationType.TABLE, generator = "concertGenerator")
   @Column(name = "ID")
   private Long id;
```

Generated Identifiers - Identity

```
@Entity
@Table(name = "CONCERT")
public class Concert implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Long id;
```

Identifiers - Compound

- Meerdere primary keys

Identifiers - Compound

- Meerdere primary keys

```
@Entity
public class Concert {

    @Id
    private String artist;

    @Id
    private Date date;
```

Identifiers - Compound

- Meerdere primary keys

```
@Entity
@IdClass(ConcertId.class)
public class Concert {

    @Id
    private String artist;

    @Id
    private Date date;
```

```
public class ConcertId {

    private String artist;
    private Date date;
```

Identifiers - Embeddable

```
@Entity
public class Concert {

   @EmbeddedId
   private ConcertId concertId;
```

```
@Embeddable
public class ConcertId {

   private String artist;
   private Date date;
```

Identifiers - Derived

- Foreign Key in Primary Key

```
@Entity
public class Concert {

   @Id
   @JoinColumn(name = "artist_id",
          referencedColumnName = "id")
   private Artist artist;

   @Id
   private Date date;
```

Identifiers - Derived

- Foreign Key in Primary Key

```
@Entity
public class Concert {

   @Id
   @JoinColumn(name = "artist_id",
          referencedColumnName = "id")
   private Artist artist;

   @Id
   private Date date;
```

- Foreign Key = Primary Key

```
@Entity
public class AccountInfo {

   @Id
   @OneToOne
   @JoinColumn(name = "account_id",
          referencedColumnName = "id")
   private Account account;
```

Entity Mapping - Enums

```
@Entity
@Table(name = "ARTIST")
public class Artist implements Serializable {

  @Enumerated(EnumType.STRING)
  @Column(name = "GENRE")
  private Genre genre;
```

Entity Mapping - Element Collection

```
@Entity
public class Artist {

    @Id
    private Long id;
    private String name;
    private List<String> aliases;
```

Entity Mapping - Element Collection

```
@Entity
public class Artist {

    @Id
    private Long id;
    private String name;
    private Set<String> aliases;
```

```
@Entity
public class Artist {

    @Id
    private Long id;
    private String name;
    @ElementCollection
    @CollectionTable(name=”AristAlias”,
        joinColumn=@JoinColumn(name=”Artist_Id”))
    private Set<String> aliases;
```

Entity Mapping - Map Mapping

```
@Entity
@Table(name = "Artist")
public class Artist implements Serializable {


  @OneToMany(mappedBy=”artist”)
  @MapKey(name=”id”)
  private Map<Long, Concert> concerts;
```

Entity Mapping - Map Mapping

```
@Entity
@Table(name = "Artist")
public class Artist implements Serializable {

  @OneToMany(mappedBy=”artist”)
  @MapKey(name=”id”)
  private Map<Long, Concert> concerts;
```

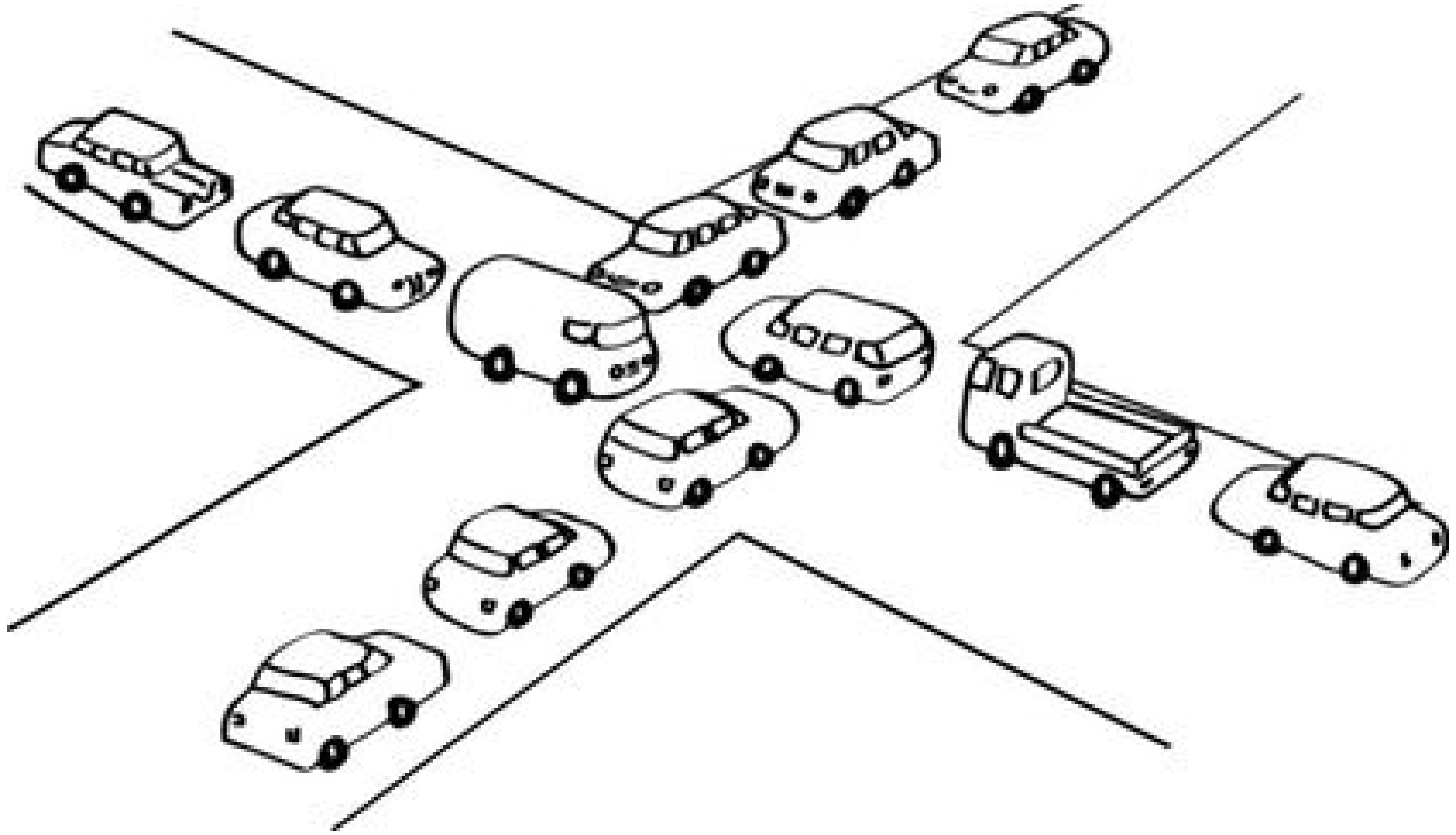
```
@Entity
@Table(name = "Artist")
public class Artist implements Serializable {

    @ElementCollection
    @CollectionTable(name=”Artist_Phone”)
    @MapKeyEnumerated(EnumType.STRING)
    @MapKeyColumn(name=”Type”)
    @Column(name=”Phone_Number”)
    private Map<PhoneType, String> phoneNumbers;
```

Inheritance

- Abstract Classes
- Single Table Strategy
- Joined Strategy
- Table per Concrete Strategy

Locking

- Read Committed
- Optimistic Read/Write Lock
- Pessimistic Read/Write Lock
  - Deadlocks
  - Scope
  - Timeouts



Performance

- Caching
- Batching
- Database power

Performance

- Caching
- Batching
- Database power

Performance - Caching

- Application Cache
- Entity Manager Cache (1st level)
- Shared Cache (2nd level)

Performance - Caching

- Application Cache
- Entity Manager Cache (1st level)
- Shared Cache (2nd level)

- Type
  - Object Cache
  - Data Cache
  - Query Cache

- Strategies
  - Full
  - Read-Only
  - LRU

Performance - Batching

- Database in: Batch Inserts

- Hibernate Properties
  - hibernate.jdbc.batch_size
  - hibernate.order_inserts
  - hibernate.order_updates

- Let op!

Performance - Batching

- Database uit: Batch Retrieval

- Hibernate ScrollableResultSet

Performance - Batching

- Database uit: Batch Retrieval

- Hibernate ScrollableResultSet

```
Session session = em.unwrap(Session.class);
sessionFactory = session.getSessionFactory();
statelessSession = sessionFactory.openStatelessSession();
Query query = statelessSession.createQuery("select tx from
DeviceTransaction tx");
query.setCacheable(false);
scrollableResults = query.scroll(ScrollMode.FORWARD_ONLY);
while (scrollableResults.next()) {
    DeviceTransaction tx = (DeviceTransaction) scrollableResults.get(0);
}
scrollableResults.close();
```

Performance - Database power

- Database Development
- Vendor Native SQL
- Waarom niet?

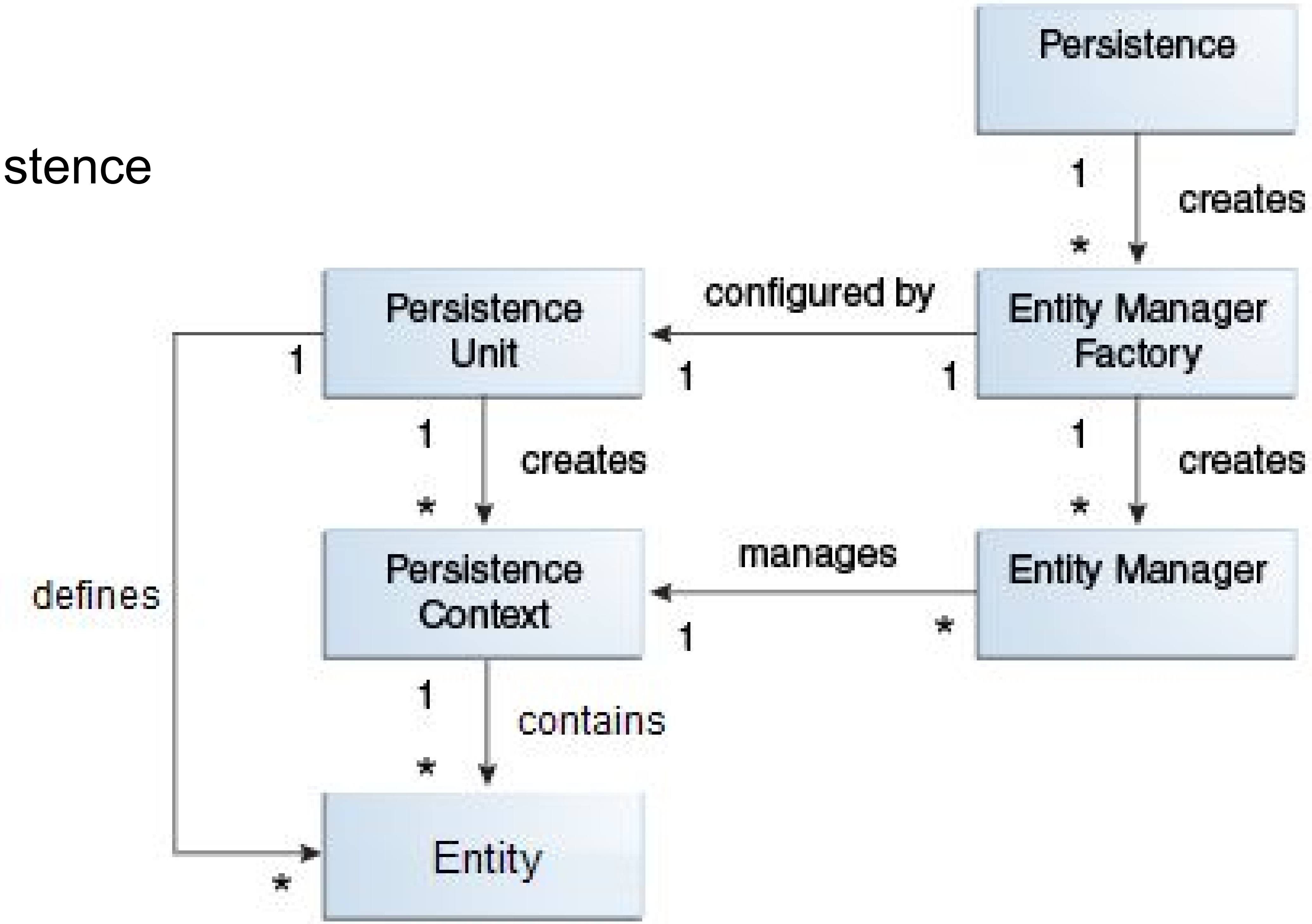
- Stored Procedures
- Hints

Security

- SQL Injection?
- Database rechten
- Gescheiden Data Sources & Persistence Units

Security

- SQL Injection?
- Database rechten
- Gescheiden Data Sources & Persistence Units



Vragen?

Op zoek naar een leuke stageplek binnen First8 of Conclusion?

Neem contact op met Bart Koekman
bkoekman@conclusion.nl
06-53300473

FIRST8