

Hibernate Envers

Entitet Versjonering i Hibernate

Sten Aksel Heien

know it

(Arendal)

**Presentasjonen er hovedsakelig en
fornorsking av presentasjonen**

”Envers – Easy Entity Auditing”

laget av Adam Warski – skaperen av Envers

<http://www.jboss.org/envers/downloads/presentations/>

© Adam Warski; adam@warski.org
(blog) - <http://www.warski.org>

Agenda

- Introduksjon
- Mønstre for temporær informasjon
- Hva er Envers?
- Hvordan virker det?
- Konfigurasjon
- Eksempel
- Prosjekt status
- Oppsummering

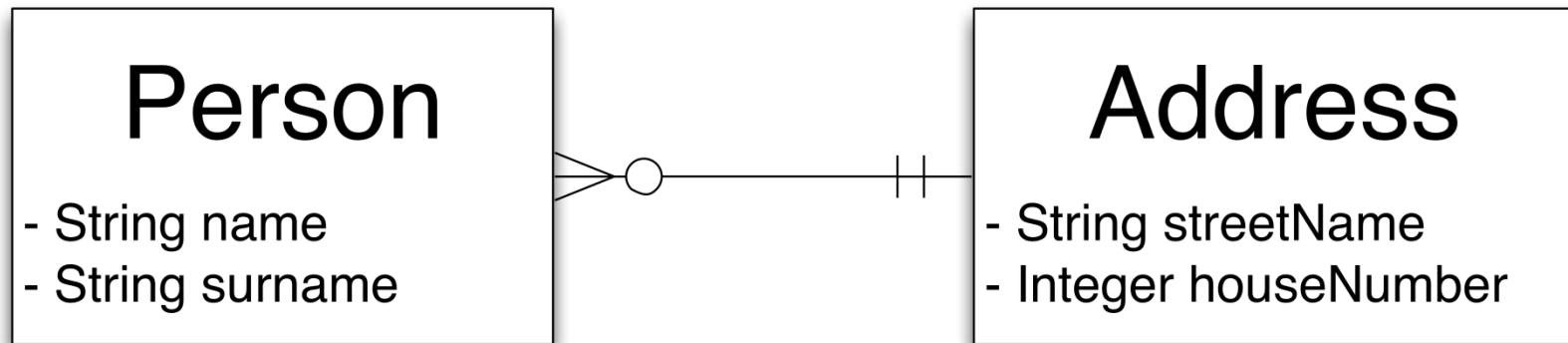
Introduksjon – første møte

- Versjonering og/eller logging av endringer på persisterte entiteter er ofte et krav
 - Post- og Teletilsynet: ifm. nummerforvaltning
- Vi sjekket JBoss Envers (beta) sommeren 2008
 - Envers var “uferdig”, og manglet bl.a. versjonering av collections
 - Vi implementerte derfor versjonering selv (... “90% ferdig”)

Introduksjon - eksempel: Person & Adresse

Problemstilling:

Vi vil lagre adressehistorikk for person



Mønster: Endringslogg

<http://martinfowler.com/eaDev/AuditLog.html>

- Logge all aktivitet til en fil eller database
 - Format: tekst, CSV eller XML for komplekse strukturer
- Simpelt (men effektivt)
- Vanskelig å gjenfinne historiske data

```
2008-08-10 06:24:19,761 [org.jboss.envers.example.Person] add,id=1,name=John,surname=Doe,address=10
2008-08-25 18:49:18,420 [org.jboss.envers.example.Person] mod,id=1,name=John,surname=Doe,address=24
2008-09-10 13:44:42,120 [org.jboss.envers.example.Address] add,id=29,streetName=East st.,houseNumber=53
2008-09-16 01:12:33,590 [org.jboss.envers.example.Person] mod,name=John,surname=Doe,address=29
```

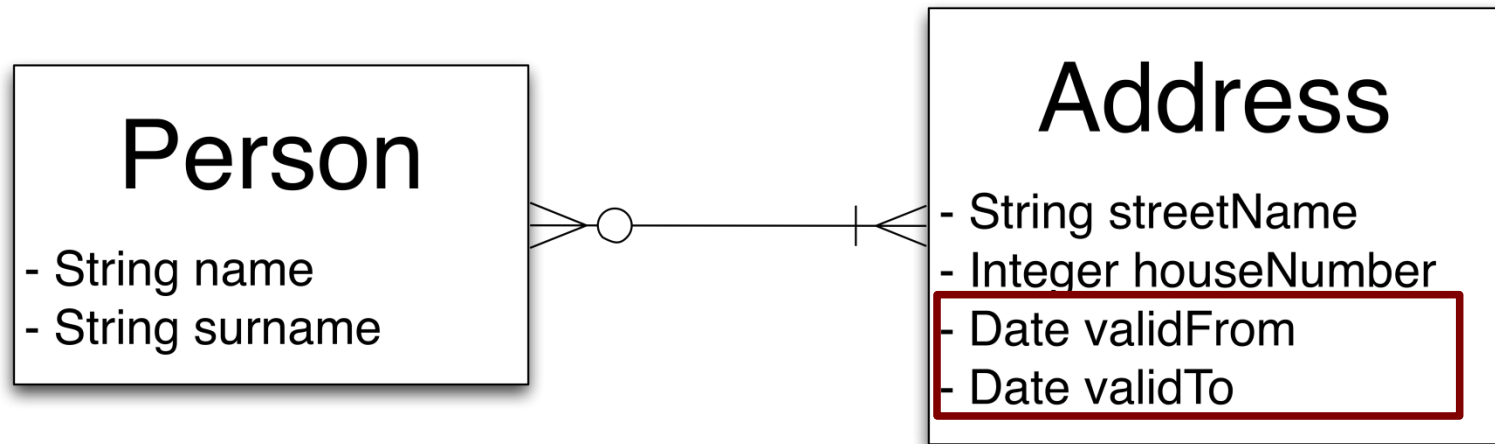
█
~

know it

Mønster: Gyldighet

<http://martinfowler.com/eaDev/Effectivity.html>

- Merke entitet med eksplisitt start- og sluttdato
- "Forurensner" domenemodellen
- Manuell håndtering
- Kompliserer mapping

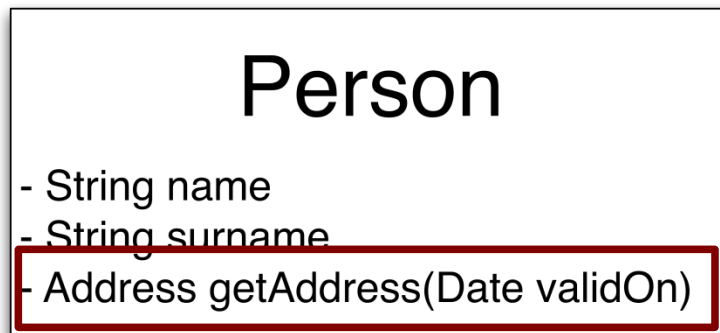


know it

Mønster: Temporær egenskap

<http://martinfowler.com/eaDev/TemporalProperty.html>

- Getter med dato argument
- Kompliserer mapping; facade må benyttes
- Verbos
- Om flere egenskaper er temporærer:
temporært objekt / snapshot



Hva er Envers?

- Et enkelt **entitet versjoneringsbibliotek**
- For JPA og Hibernate
- Forenkler lagring og gjenfinning av historiske data

Hvordan virker Envers?

- Utvikler spesifiserer hva som skal versjoneres – *entitet(er)* / *egenskap(er)* og/eller *relasjon(er)* annoteres med **@Audited**
- For hver versjonert entitet, vil det finnes en ”**revisjonsentitet**” med tilhørende ”**revisjonstabell**”
 - f.eks. entiteten “Address” har en “Address_AUD” “revisjonstabell” (postfix kan endres)
- Ifm. CRUD-operasjonene insert/update/delete vil en ny ”**revisjonsentitet**” opprettes og legges inn i revisjonstabellen

Hvordan virker Envers?

- "Revisjonsentiteten" inneholder utført operasjon med tidspunkt, revisjonsnummer og de aktuelle data
- Alle endringer i en transaksjon vil tilhøre samme revisjon = få samme revisjonsnummer
- Revisionene er globale
 - Prinsippet likner Subversion
- Med et revisjonsnummer kan du nå enkelt finne
 - alle entiteter med denne revisjonen, eller
 - dato for når revisjonen ble gjort

Person & Person_AUD

Person:

| Id | Name | Surname |
|-----|------|---------|
| 123 | John | Smith |
| 857 | Brad | Pitt |
| 698 | Mary | Doe |
| ... | | |

Person_AUD:

| Rev number | Id | Name | Surname | Rev type |
|------------|-----|--------|----------------|------------|
| 64 | 123 | James | Smith | ADD |
| 64 | 857 | Brad | Pitt | ADD |
| 85 | 123 | Peter | Smith | MOD |
| 90 | 501 | Arnold | Schwarzenegger | DEL |
| 90 | 123 | John | Smith | MOD |

Envers konfigurasjon

- Annoter med **@Audited** :

```
@Entity
public class Person {
    @Id
    @GeneratedValue
    private int id;

    @Audited
    private String name;

    @Audited
    private String surname;

    @Audited
    @ManyToOne
    private Address address;
```

- Legg inn **event listeners**
i persistence.xml

```
<property name="hibernate.ejb.event.post-insert"
value="org.hibernate.envers.event.AuditEventListener" />
<property name="hibernate.ejb.event.post-update"
value="org.hibernate.envers.event.AuditEventListener" />
<property name="hibernate.ejb.event.post-delete"
value="org.hibernate.envers.event.AuditEventListener" />
<property name="hibernate.ejb.event.pre-collection-update"
value="org.hibernate.envers.event.AuditEventListener" />
<property name="hibernate.ejb.event.pre-collection-remove"
value="org.hibernate.envers.event.AuditEventListener" />
<property name="hibernate.ejb.event.post-collection-recreate"
value="org.hibernate.envers.event.AuditEventListener" />
```

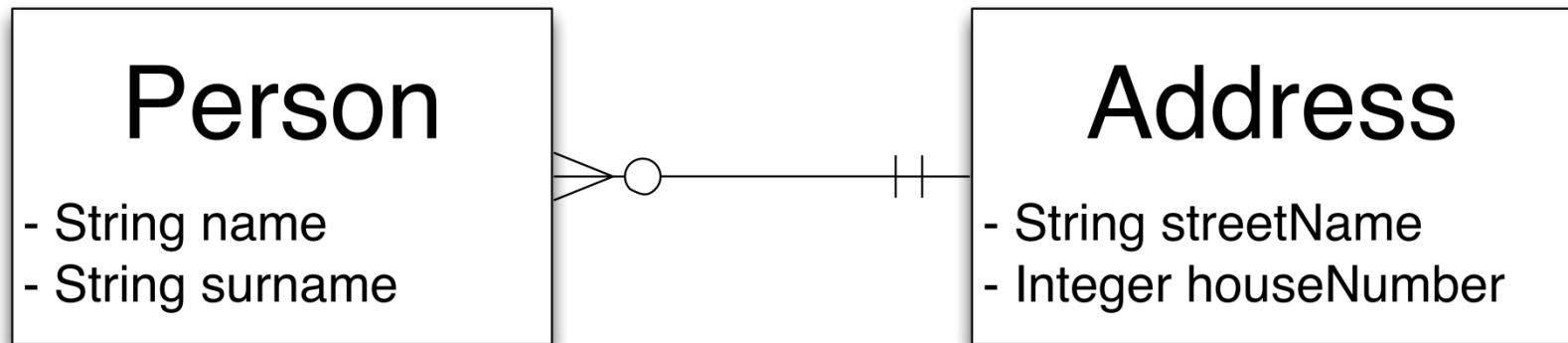
know it

Hva kan versjoneseres?

- Mapping definert av JPA:
 - Egenskaper: Strings, Integers, Dates, ...
 - Komponenter (Components)
 - Relasjoner
- Noen Hibernate utvidelser
 - Egendefinerte typer (Custom types)
 - Collections

Person & Adresse eksempelet igjen

- Anta nå at begge entitetene har blitt annotert med `@Audited`



```
// Revisjon 1

em.getTransaction().begin();

Address adr1 = new Address("Storgt.", 1);

Address adr2 = new Address("Havnegt.", 2);

Person p = new Person("Ola", "Nordmann");

p.setAddress(adr1);

entityManager.persist(adr1);

entityManager.persist(adr2);

entityManager.persist(p);

em.getTransaction().commit();
```

```
// Revisjon 2
```

```
em.getTransaction().begin();
```

```
p = entityManager.find(Person.class, id);
```

```
p.setName("Per");
```

```
p.setAddress(adr2);
```

```
em.getTransaction().commit();
```

- Nye data i entiteten – etter commit av revisjon 2
- Gamle data finnes i revisjon 1
- Ingen kodeendringer – helt transparent for bruker

```
assert "Per".equals(p.getName());  
assert adr2.equals(p.getAddress());
```

```
// Lese person i revisjon 1:
```

```
AuditReader ar = AuditReaderFactory.get(em);
```

```
x_p = ar.find(Person.class, id, 1);
```

```
assert "Ola".equals(x_p.getName());
```

```
assert adr1.equals(x_p.getAddress());
```

- Transparent traversering av relasjoner

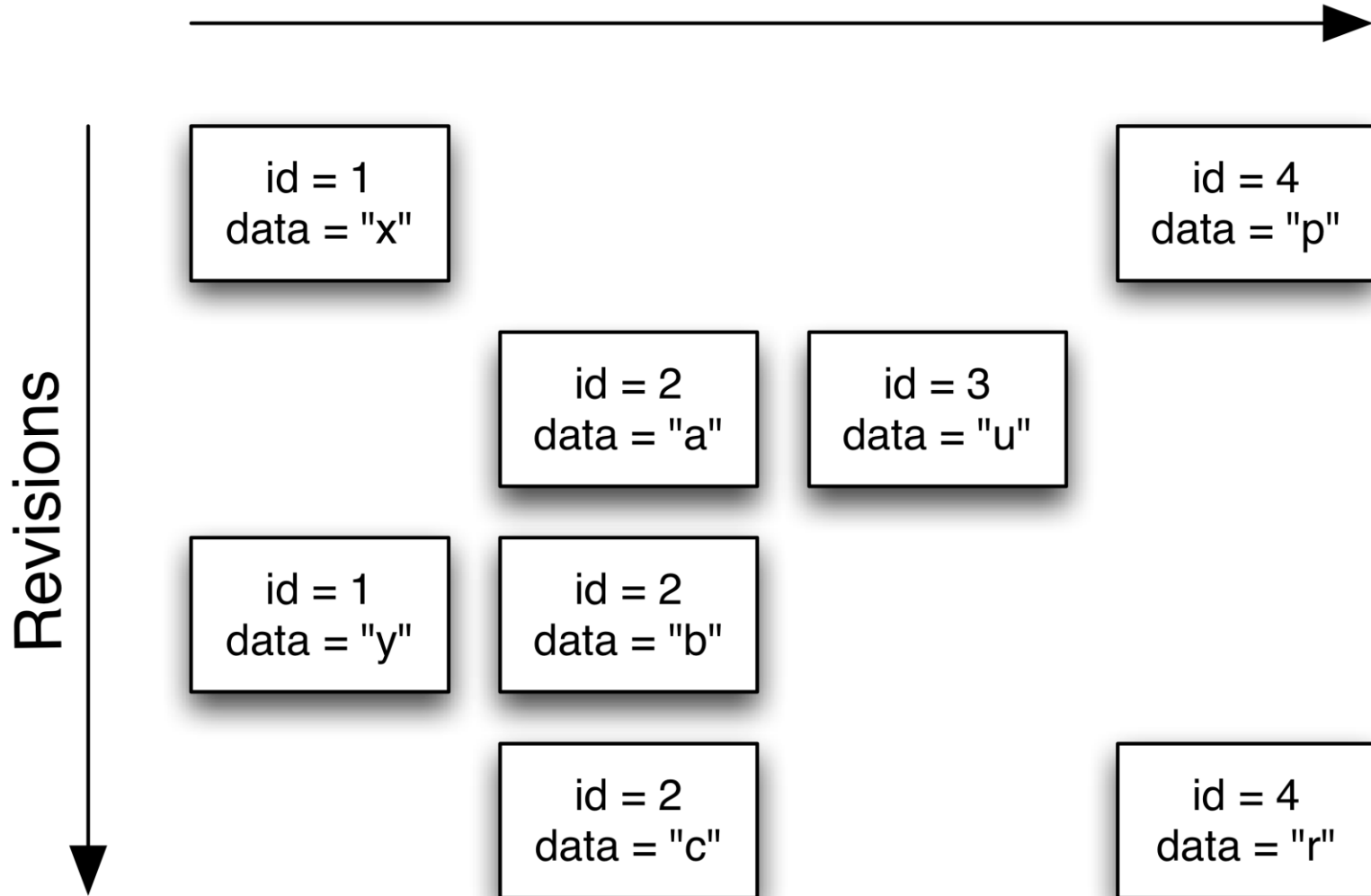
```
// Lese adressene i revisjon 1
x_adr1 = auditReader.find(Address.class, adr1_id, 1);
assert x_adr1.getPersons().size() == 1;
assert x_adr1.getPersons().contains(p);

x_adr2 = auditReader.find(Address.class, adr2_id, 1);
assert x_adr2.getPersons().size() == 0;
```

- Transparent traversering av relasjoner, også collections

Hva kan det spørres etter?

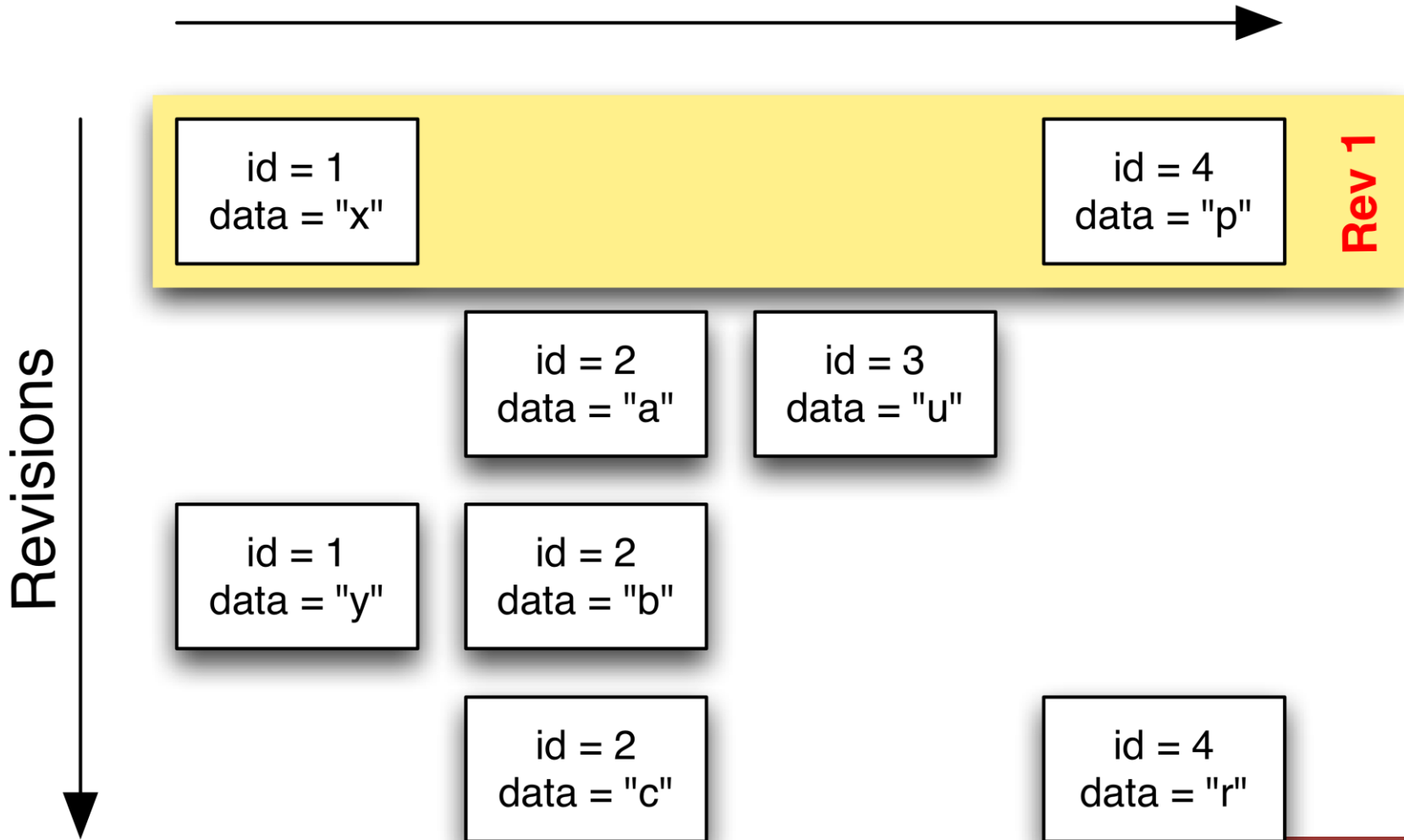
Entities



know it

Hva kan det spørres etter? Entiteter-i-revisjon

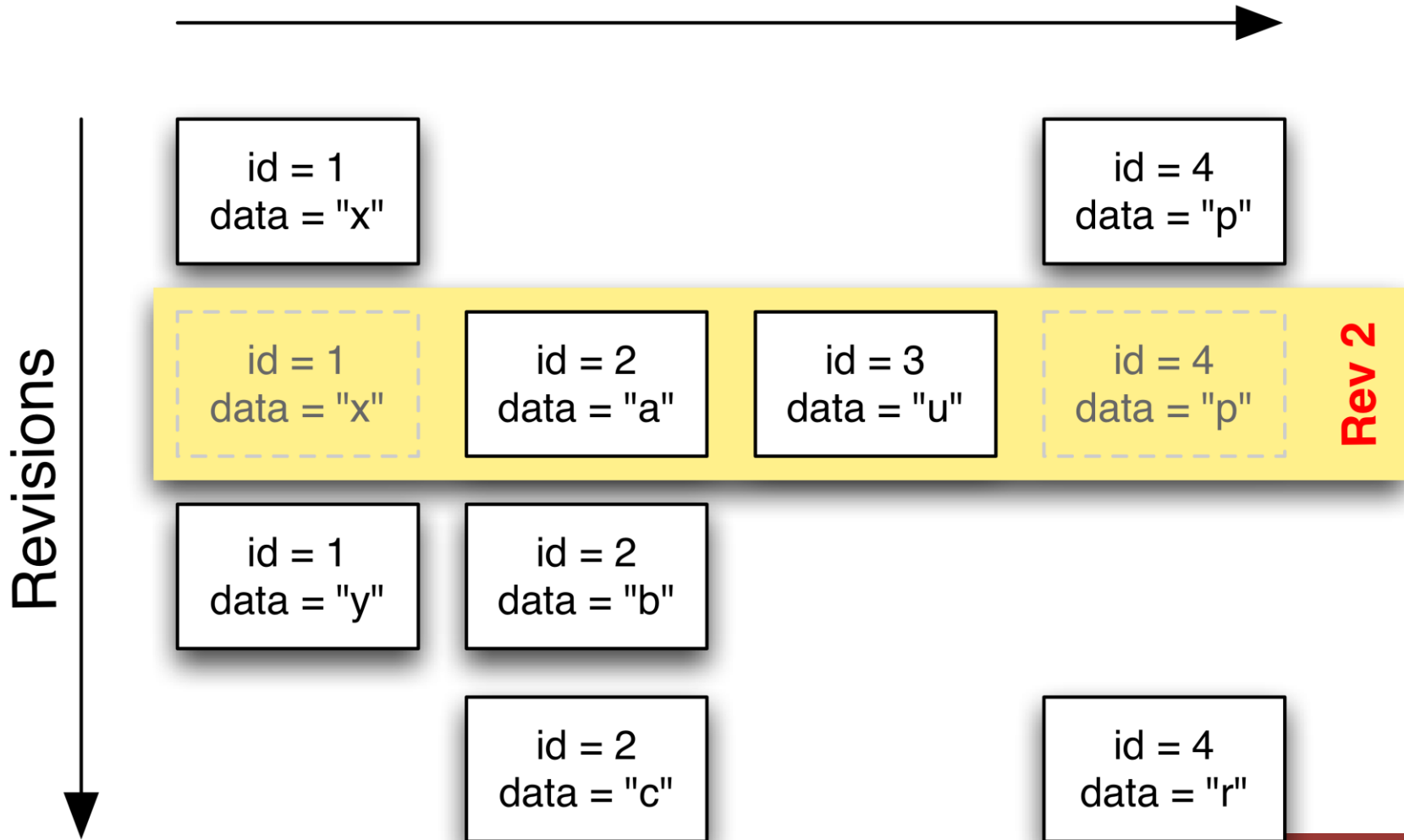
Entities



know it

Hva kan det spørres etter? Entiteter-i-revisjon

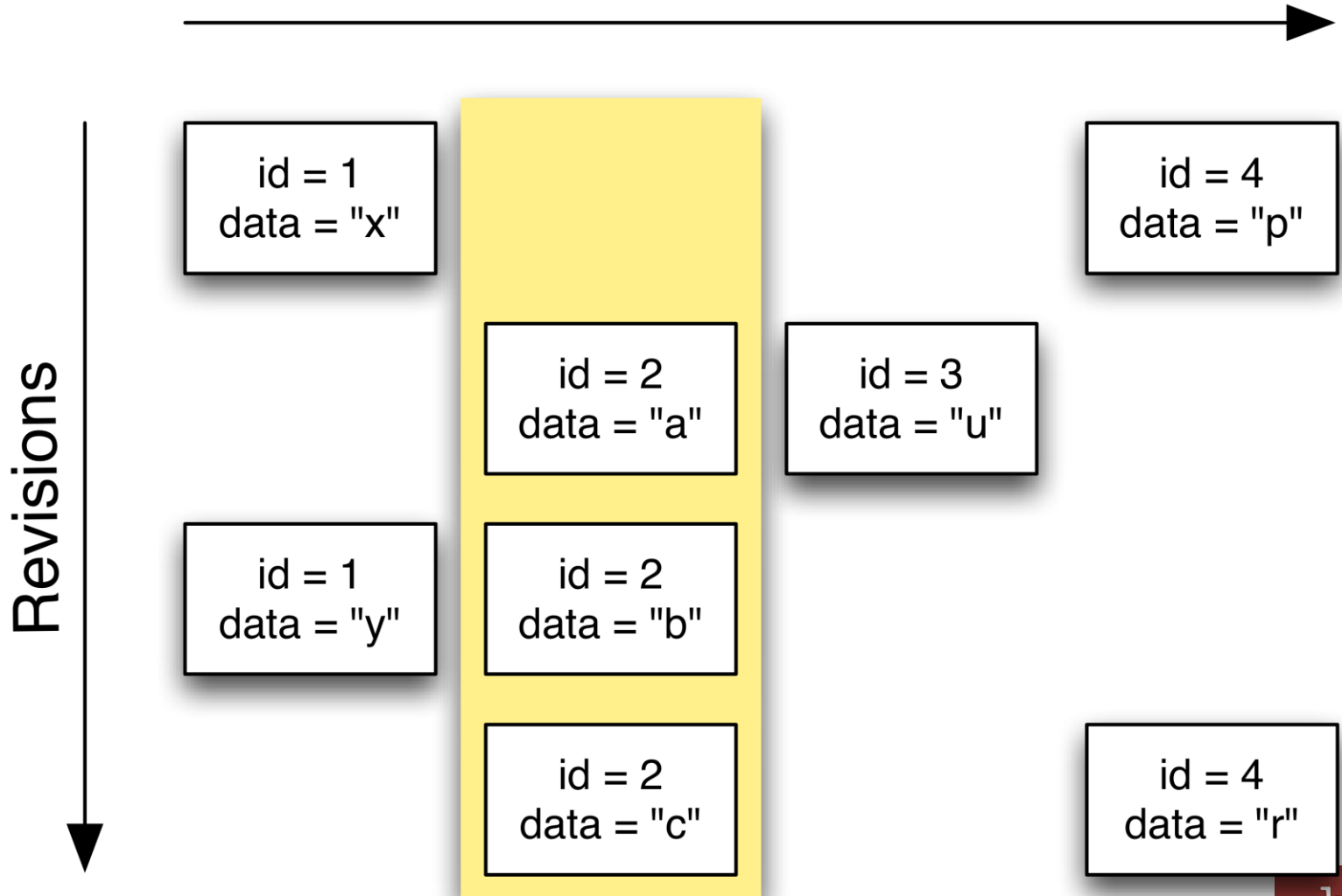
Entities



know it

Hva kan det spørres etter? Revisjon-av-entitet

Entities



know it

Hva kan det spørres etter?

- Entiteter-i-revisjon (forEntitiesAtRevision)
- Revisjon-av-entitet (forRevisionsOfEntity)
- Inspirert av kriteria spørringer (criteria queries)

```
auditReader.createQuery()  
  .forRevisionsOfEntity(Person.class, false, true)  
  .addProjection(AuditEntity.revisionNumber().count())  
  .add(AuditEntity.id().eq(person.getId()))  
  .getSingleResult()
```

Logging av metadata for revisjon

- Metadata kan knyttes til en revisjon
 - F. eks.: bruker som utfører endringer
- En spesial "revisjonsentitet" (`@RevisionEntity`), lagrer metadataene
- Forskjellige "Listeners", blir kalt når en ny entitet lages, oppdateres eller slettes

Eksempel : Wiki

- "Wiki power": alle kan redigere
- Historikk skal lagres og kunne sees av alle
- Pluss, man kan se hvem som endret siden

@Audited entity

```
@Entity
@Audited
public class WikiPage {

    @Id @GeneratedValue
    private Long id;

    private String name;

    private String description;

    @OneToMany
    private Set<Link> links;

    @OneToMany
    private Set<Image> images;

    // Getters, setters, equals, hashCode ...
}
```

@RevisionEntity

```
@Entity
@RevisionEntity(WikiListener.class)
public class WikiRevision {

    @Id @GeneratedValue
    @RevisionNumber
    private Long id;

    @RevisionTimestamp
    private Long timestamp;

    @OneToMany
    private User modifiedBy;

    // Getters, setters, equals, hashCode ...

}
```

RevisionListener

```
public class WikiListener implements RevisionListener {  
  
    public void newRevision(Object revEntity) {  
  
        WikiRevision wikiRev = (WikiRevision) revEntity;  
  
        User currentUser = (User) Component.getInstance  
            ("currentUser");  
  
        wikiRev.setModifiedBy(currentUser);  
    }  
}
```

Paginert spørring i historikken

```
public List getHistory(int from, int count, Long pageId) {  
  
    return auditReader.createQuery()  
        .forRevisionsOfEntity(WikiPage.class, false, true)  
        .addOrder(revisionNumber.desc())  
        .add(id(pageId))  
        .setFirstResult(from)  
        .setMaxResults(count)  
        .getResultList();  
}
```

I databasen:

Page:

| Id | Title | Content |
|-----|---------|-----------|
| 123 | Title 1 | Content 1 |
| 857 | Title 2 | Content 2 |
| 698 | Title 3 | Content 3 |
| ... | | |

Page_AUD:

| Rev number | Id | Title | Content | Rev type |
|------------|-----|----------------|------------------|------------|
| 64 | 123 | Oldest title 1 | Oldest content 1 | ADD |
| 64 | 857 | Title 2 | Content 2 | ADD |
| 85 | 123 | Older title 1 | Older content 1 | MOD |
| 90 | 501 | Title 4 | Content 4 | DEL |
| 90 | 123 | Title 1 | Content 1 | ADD |

Envers historikk

- 16. juli 2008: JBoss Envers 1.0.0.GA
- 21. okt. 2008: JBoss Envers 1.1.0.GA
 - Vi så igjen på Envers ... 😊
 - Alle JPA collection mappinger støttes
- 31. okt. 2008 Hibernate liker også Envers
 - Envers flyttes til Hibernate ... “og vi er ikke i tvil”
- 5. mars 2009: Hibernate Envers 1.2.0.GA

NB! Annotering byttes ☹

- 16. juli 2008: JBoss Envers 1.0.0.GA
- 21. okt. 2008: JBoss Envers 1.1.0.GA

~~@Versioned~~

- 5. mars 2009: Hibernate Envers 1.2.0.GA

@Audited

Ytelse

- Tregere enn uten versjonering:
 - 1 insert for hver endret entitet
 - 1 insert for hver transaksjon

| MySQL 5.1.30 (InnoDB) | 5000 inserts | 1000 complex inserts | 5000 updates |
|--------------------------|-----------------|-------------------------|-----------------|
| Not audited | 6.307s | 6.622s | 8.487s |
| Audited | 9.807s | 12.758s | 11.444s |
| Difference | x 1.55 | x 1.92 | x 1.34 |

Envers oppsummering

- Uendret **mapping** (not intrusive)
- Uendret **kode** (transparent)
- Enkel **historie lesing**
- **Slettede** entiteter er ikke borte
- **Enkelt** å bruke (**@Audited**)
- Data for revisions
- **Kraftige spørringer**

Rykter om videre funksjonalitet

- Verktøy
 - Import
 - Revert
 - Branch
- Diff
- Forskjellige strategier for revisjonshåndtering
- Tagging
 - Find entity-by-tag

Når passer Envers (best)?

- Stort sett interessert i “de gyldige” dataene mer sjeldent skal de historiske data gjenfinnes
- Ingen ”over-hyppige” endringer av data

Referanser

- "Data Versioning and Envers", Adam Warski
Takk til Adam - skaperen av Envers - for å ha sendt meg engelsk presentasjon; adam@warski.org
- Adam Warski (blog) - <http://www.warski.org>
- <http://www.jboss.org/envers/>
- <http://jboss.org/files/envers/docs/index.html>
- Patterns of Enterprise Application Architecture:
 - <http://martinfowler.com/eaDev/index.html>

Hibernate Envers



by Sten Aksel Heien

know it

(Arendal)