Schema-Based Security in Neo4j 4.0

Louise Söderström

neo4j

About me

- Cypher and security developer @ Neo4j since 2017
- Was born in the early 90s in Linköping, Sweden
- Engineer in Mathematics
- Involved in Pink Programming

Native security in Neo4j 3.x

- Based on files for users and user-to-role mapping
- Only coarse-grained built-in roles:
  - reader,editor, publisher, architect, admin
- Managed via procedures

Schema-based security in Neo4j 4.0

- The security data is stored in a system database
- Fine-grained access, traverse and read
- Coarse-grained writes
- Managed via administration commands

How to use the system database

- Browser/desktop/Cypher shell **`:use system`**
- Drivers
  - Supported for Java, JS and .NET
  - Session construction methods take optional name argument

```
try ( Session s1 = driver.session(forDatabase("system")) {
   s1.run( "GRANT ROLE reader TO currentUser" );
}
try (Session s2 = driver.session() ) {
   s2.run( "MATCH (n) RETURN n.prop" );
}
```

User and role administration

- 3.x security with Cypher instead of procedures
- Users in community, roles only in enterprise
- Old security procs will still work but must be executed towards system database,
  except **`dbms.security.changePassword()`**
  Note: for now, **`yield`** will not be supported

Create users

* **STATUS** is not available in community

- `CREATE USER Alice SET PASSWORD $secret`
- `CREATE USER Bob SET PASSWORD $secret2`
  `CHANGE NOT REQUIRED`
- `CREATE USER Charlie SET PASSWORD $secret3`
  `SET STATUS SUSPENDED*`

Change users*

* Not available in community except `ALTER CURRENT USER`

- `ALTER USER Alice SET PASSWORD CHANGE NOT REQUIRED`
- `ALTER USER Bob SET PASSWORD $anotherSecret`
- `ALTER USER Charlie SET STATUS ACTIVE`
- `ALTER CURRENT USER SET PASSWORD FROM old TO new`

Create and delete roles

- Create role:

```
CREATE ROLE employee
CREATE ROLE doctor
CREATE ROLE receptionist
CREATE ROLE researcher
CREATE ROLE dummy
```

- Delete role:

```
DROP ROLE dummy
```

Grant and revoke roles

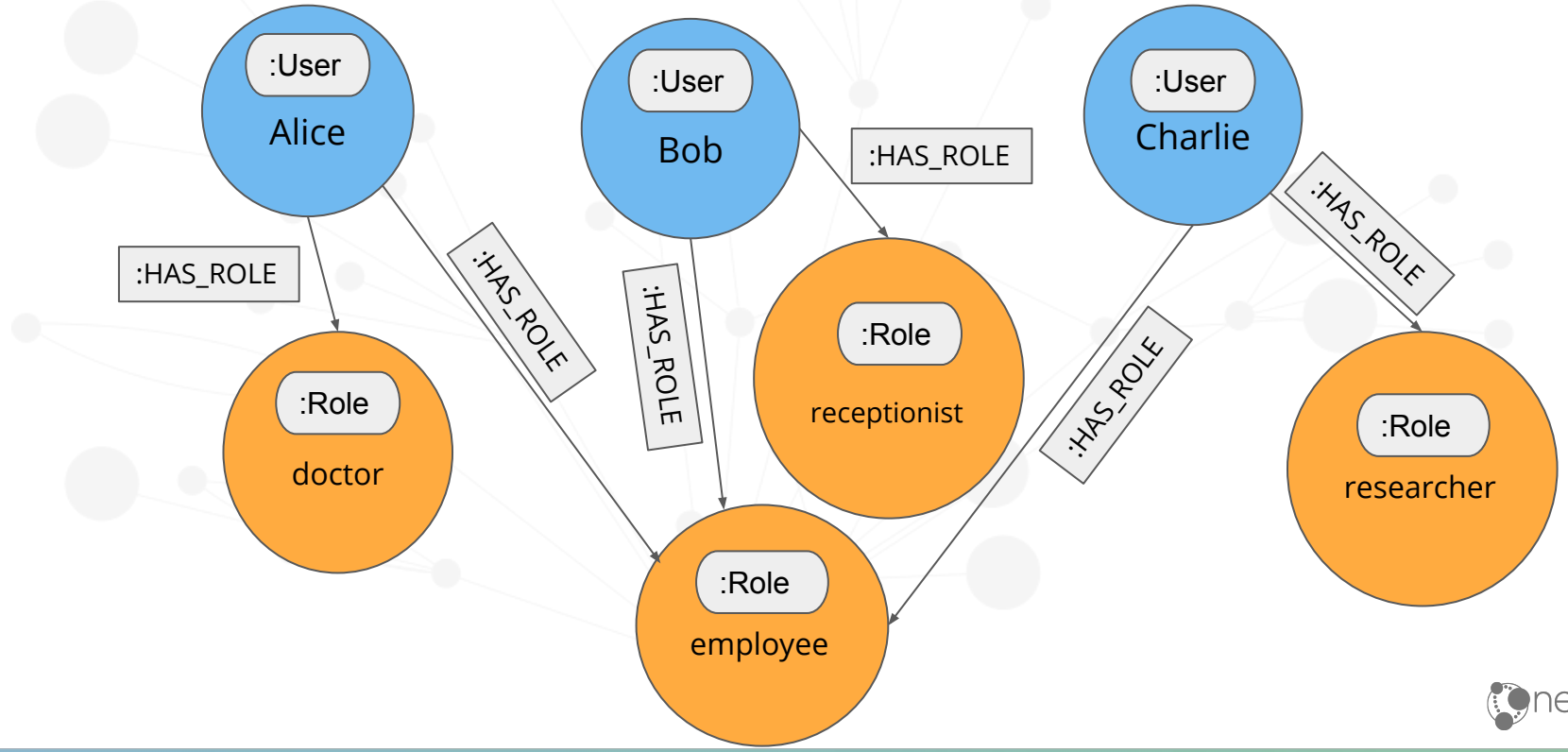
- Grant role to user:

```
GRANT ROLE employee TO Alice, Bob, Charlie
GRANT ROLE doctor, researcher TO Alice
GRANT ROLE receptionist TO Bob
GRANT ROLE researcher TO Charlie
```

- Revoke role from user:

```
REVOKE ROLE researcher FROM Alice
```

Users and roles



Privilege administration

- Fine-grained 4.0 security
- Each role have privilege whitelist (**`GRANT`**) and privilege blacklist (**`DENY`**), privileges can be unassigned by **`REVOKE`**
- Aggregated permission: a user is allowed to do an action if it is in at least one whitelist and no blacklist

Privileges

- **`ACCESS`** – access for specific dbs
- **`TRAVERSE`** – node/relationship traversal for specific dbs and/or labels/reltypes
- **`READ`** – property reads for specific dbs and/or labels/reltypes and/or property key names
- **`MATCH`** - shorthand for **`TRAVERSE`** + **`READ`**
- Write, token, indexes, constraints, start and stop databases

Access privilege

- **`GRANT ACCESS ON DATABASE`** *`healthcare`* **`TO`** *`employee`*

- With only access a user with role *`employee`* will
  - be able to run read queries but get an empty result
  - get **`PERMISSION DENIED`** on write queries

- On a database where a user has no access, she will get **`PERMISSION DENIED`** on transaction start

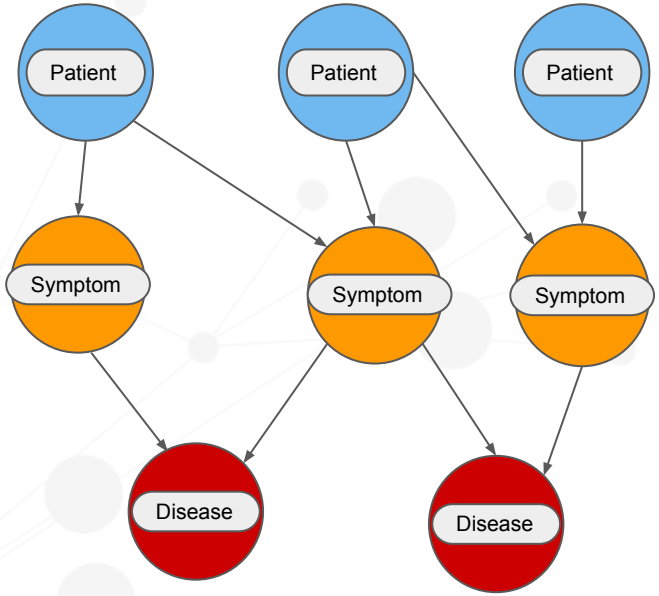
Alice the doctor

```
GRANT TRAVERSE ON GRAPH
healthcare TO doctor
```

```
GRANT READ {*} ON GRAPH
healthcare TO doctor
```

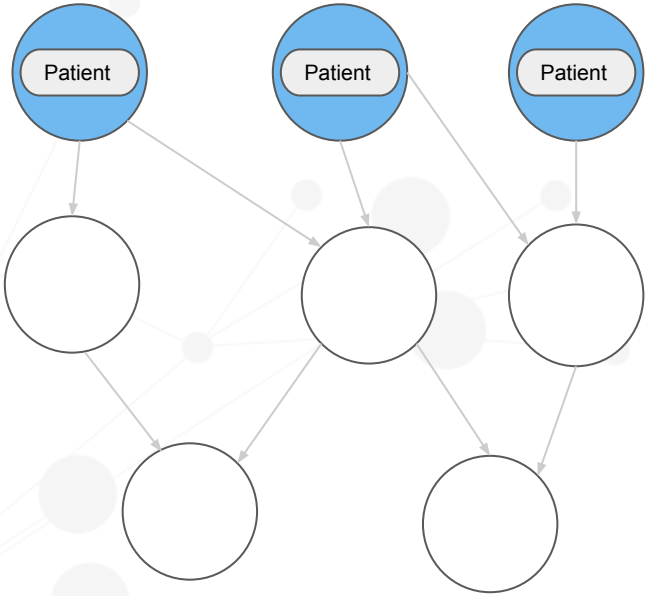
OR

```
GRANT MATCH {*} ON GRAPH
healthcare TO doctor
```



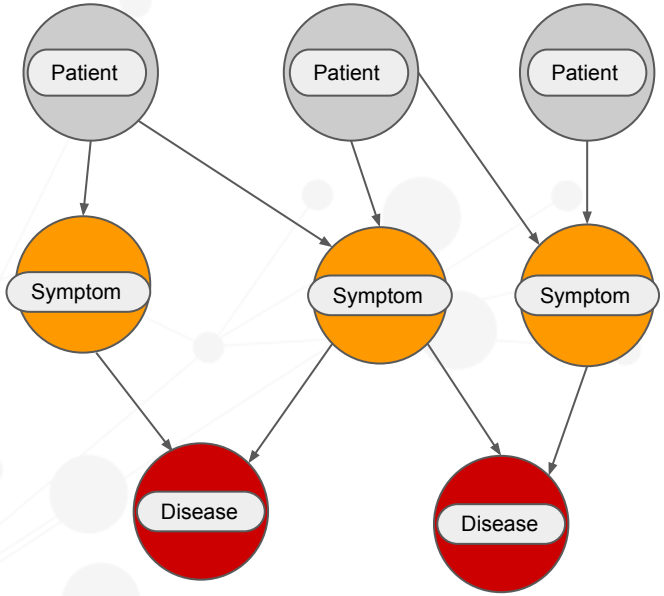
Bob the receptionist

```
GRANT MATCH {*} ON GRAPH
healthcare NODES Patient TO
receptionist
```



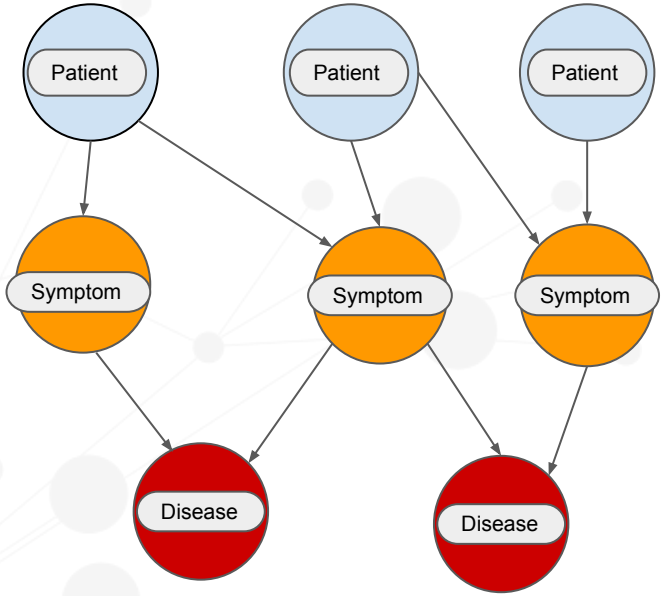
Charlie the researcher - approach 1

```
GRANT TRAVERSE ON GRAPH
healthcare TO researcher

GRANT READ {*} ON GRAPH
healthcare NODES Symptom TO
researcher

GRANT READ {*} ON GRAPH
healthcare NODES Disease TO
researcher
```



Charlie the researcher - approach 2

```
GRANT MATCH {*} ON GRAPH
healthcare TO researcher

DENY READ {name, ssn} ON
GRAPH healthcare NODES
Patient TO researcher
```



Write and token privileges

```
GRANT WRITE {*} ON GRAPH healthcare TO doctor
GRANT WRITE {*} ON GRAPH healthcare TO receptionist
DENY  WRITE {*} ON GRAPH * TO researcher

GRANT CREATE NEW NODE LABEL ON DATABASE
healthcare TO doctor
GRANT CREATE NEW PROPERTY NAME ON DATABASE
healthcare TO doctor
```

Indexes, constraints and more

```
GRANT INDEX MANAGEMENT ON DATABASE healthcare TO researcher

GRANT CREATE CONSTRAINT ON DATABASE healthcare TO doctor

GRANT ALL DATABASE PRIVILEGES TO superAdmin
```


Thanks for your time!
Questions?
neo4j

Hunger Games Questions

1. Easy: What is **`MATCH`** a combination of?
   a. **`ACCESS`** + **`TRAVERSE`**
   b. **`ACCESS`** + **`READ`**
   c. **`TRAVERSE`** + **`READ`**

2. Medium: What will happen to a user who doesn't have the access privilege?
   a. **`PERMISSION DENIED`** at transaction start
   b. The transaction starts but **`PERMISSION DENIED`** when reading
   c. Reading will work but give an empty result

3. Hard: Which procedure is not going to work in 4.0?