Database Management Systems
Rel DB Recitation

MEHMET TAHİR SANDIKKAYA, 14 OCTOBER 2015, ISTANBUL TECHNICAL UNIVERSITY.

Content

- Installation & Running
- Examples
- Database design
- Relational algebra

Installation

- Make sure to successfully install a recent version of Java
  - JDK is better than JRE
- Download the RelDB installation jar file
- Run it with root/admin rights
  - You may need to open a console with admin rights in Windows and type “java –jar RelinstallXX.jar”
- After the installation, run the DBrowser with admin rights (for DB operations on the file system)
  - Similarly, you may need a privileged console to run “java –jar Dbrowser.jar”

Examples

- Defining a type

```
TYPE SCORE POSSREP
  {VALUE RATIONAL
    CONSTRAINT (VALUE >= 1.0) AND (VALUE <= 10.0)
  };
```

  - Take care about expressions and statements

- After a type is defined, a value of that type can be expressed

```
SCORE(6.7);
```

  - Note that, you cannot state anything by expressing a score value

```
SCORE(0.7)
```

  - You cannot violate the constraint

Examples

- Use THE_ to obtain the primitive value from a type variable

```
THE_VALUE(SCORE(9.7))
```

- Use CAST_AS_ to cast a variable

```
CAST_AS_INTEGER(10.9)
```

Database design

- Types (that may be special in the design) are generated

```
TYPE MOVIE# POSSREP
  {VALUE INTEGER};
```

```
TYPE YEAR POSSREP
  {VALUE INTEGER};
```

```
TYPE PERSON# POSSREP
  {VALUE INTEGER};
```

  - The score type is already defined

Database design

- Now, a relation can be formed

```
RELATION
  {MOVIE# MOVIE#,
   TITLE CHAR,
   YEAR YEAR,
   SCORE SCORE,
   VOTES INTEGER,
   DIRECTOR# PERSON#}
```

  - But, a relation is a concept
  - It cannot neither be expressed nor stated in D

Database design

- Each relation must be assigned to a variable in a statement

```
VAR MOVIE BASE RELATION
  {MOVIE# MOVIE#,
   TITLE CHAR,
   YEAR YEAR,
   SCORE SCORE,
   VOTES INTEGER,
   DIRECTOR# PERSON#}
KEY {MOVIE#};
```

  - Each relation variable **always** requires a key

Database design

- Complete the other relation variables

```
VAR PERSON BASE RELATION
  {PERSON# PERSON#,
   NAME CHAR}
KEY {PERSON#};
```

```
VAR CASTING BASE RELATION
  {MOVIE# MOVIE#,
   ACTOR# PERSON#,
   ORD INTEGER}
KEY {MOVIE#, ACTOR#};
```

Database design

- You may generate a tuple for one of the relations

```
TUPLE
  {MOVIE# MOVIE#(6),
   TITLE "Usual Suspects",
   YEAR YEAR(1995),
   SCORE SCORE(8.7),
   VOTES 35027,
   DIRECTOR# PERSON#(639)}
```

  - Is this an expression or a statement?
  - What happens when this code executes?
  - What happens if I add a semicolon?

Database design

- You may generate a relation with tuples in it

```
RELATION{
  TUPLE
    {MOVIE# MOVIE#(6),
     TITLE "Usual Suspects",
     YEAR YEAR(1995),
     SCORE SCORE(8.7),
     VOTES 35027,
     DIRECTOR# PERSON#(639)}
}
```

  - Still an expression, so not assigned

Database design

- Assign the *expressed* relation to the relation variable by a *statement* to store it in the database

```
MOVIE := RELATION{
  TUPLE
    {MOVIE# MOVIE#(6),
     TITLE "Usual Suspects",
     YEAR YEAR(1995),
     SCORE SCORE(8.7),
     VOTES 35027,
     DIRECTOR# PERSON#(639)}
};
```

Database design

- Now, fill in the database by entering bulk data

```
MOVIE := RELATION {
    TUPLE { MOVIE# MOVIE#(6), TITLE "Usual Suspects",
            YEAR YEAR(1995), SCORE SCORE(8.7), VOTES 35027,
            DIRECTOR# PERSON#(639) },
    TUPLE { MOVIE# MOVIE#(70), TITLE "Being John Malkovich",
            YEAR YEAR(1999), SCORE SCORE(8.3), VOTES 13809,
            DIRECTOR# PERSON#(1485) },
    TUPLE { MOVIE# MOVIE#(107), TITLE "Batman & Robin",
            YEAR YEAR(1997), SCORE SCORE(3.5), VOTES 10577,
            DIRECTOR# PERSON#(105) },
    TUPLE { MOVIE# MOVIE#(110), TITLE "Sleepy Hollow",
            YEAR YEAR(1999), SCORE SCORE(7.5), VOTES 10514,
            DIRECTOR# PERSON#(148) },
    TUPLE { MOVIE# MOVIE#(112), TITLE "Three Kings",
            YEAR YEAR(1999), SCORE SCORE(7.7), VOTES 10319,
            DIRECTOR# PERSON#(1070) },
    TUPLE { MOVIE# MOVIE#(151), TITLE "Gattaca",
            YEAR YEAR(1997), SCORE SCORE(7.4), VOTES 8388,
            DIRECTOR# PERSON#(2020) },
    TUPLE { MOVIE# MOVIE#(213), TITLE "Blade",
            YEAR YEAR(1998), SCORE SCORE(6.7), VOTES 6885,
            DIRECTOR# PERSON#(2861) },
    TUPLE { MOVIE# MOVIE#(228), TITLE "Ed Wood",
            YEAR YEAR(1994), SCORE SCORE(7.8), VOTES 6587,
            DIRECTOR# PERSON#(148) },
    TUPLE { MOVIE# MOVIE#(251), TITLE "End of Days",
            YEAR YEAR(1999), SCORE SCORE(5.5), VOTES 6095,
            DIRECTOR# PERSON#(103) },
    TUPLE { MOVIE# MOVIE#(281), TITLE "Dangerous Liaisons",
            YEAR YEAR(1988), SCORE SCORE(7.7), VOTES 5651,
            DIRECTOR# PERSON#(292) },
    TUPLE { MOVIE# MOVIE#(373), TITLE "Fear and Loathing in Las Vegas",
            YEAR YEAR(1998), SCORE SCORE(6.5), VOTES 4658,
            DIRECTOR# PERSON#(59) },
    TUPLE { MOVIE# MOVIE#(432), TITLE "Stigmata",
            YEAR YEAR(1999), SCORE SCORE(6.1), VOTES 4141,
            DIRECTOR# PERSON#(2557) },
    TUPLE { MOVIE# MOVIE#(433), TITLE "eXistenZ",
            YEAR YEAR(1999), SCORE SCORE(6.9), VOTES 4130,
            DIRECTOR# PERSON#(97) },
    TUPLE { MOVIE# MOVIE#(573), TITLE "Dead Man",
            YEAR YEAR(1995), SCORE SCORE(7.4), VOTES 3333,
            DIRECTOR# PERSON#(175) },
    TUPLE { MOVIE# MOVIE#(1468), TITLE "Europa",
            YEAR YEAR(1991), SCORE SCORE(7.6), VOTES 1042,
            DIRECTOR# PERSON#(615) },
    TUPLE { MOVIE# MOVIE#(1512), TITLE "Suspiria",
            YEAR YEAR(1977), SCORE SCORE(7.1), VOTES 1004,
            DIRECTOR# PERSON#(2259) },
    TUPLE { MOVIE# MOVIE#(1539), TITLE "Cry-Baby",
            YEAR YEAR(1990), SCORE SCORE(5.9), VOTES 972,
            DIRECTOR# PERSON#(364) }
};
```

```
PERSON := RELATION {
    TUPLE { PERSON# PERSON#(9), NAME "Arnold Schwarzenegger" },
    TUPLE { PERSON# PERSON#(26), NAME "Johnny Depp" },
    TUPLE { PERSON# PERSON#(59), NAME "Terry Gilliam" },
    TUPLE { PERSON# PERSON#(97), NAME "David Cronenberg" },
    TUPLE { PERSON# PERSON#(103), NAME "Peter Hyams" },
    TUPLE { PERSON# PERSON#(105), NAME "Joel Schumacher" },
    TUPLE { PERSON# PERSON#(138), NAME "George Clooney" },
    TUPLE { PERSON# PERSON#(148), NAME "Tim Burton" },
    TUPLE { PERSON# PERSON#(175), NAME "Jim Jarmusch" },
    TUPLE { PERSON# PERSON#(187), NAME "Christina Ricci" },
    TUPLE { PERSON# PERSON#(243), NAME "Uma Thurman" },
    TUPLE { PERSON# PERSON#(282), NAME "Cameron Diaz" },
    TUPLE { PERSON# PERSON#(292), NAME "Stephen Frears" },
    TUPLE { PERSON# PERSON#(302), NAME "Benicio Del Toro" },
    TUPLE { PERSON# PERSON#(308), NAME "Gabriel Byrne" },
    TUPLE { PERSON# PERSON#(350), NAME "Jennifer Jason Leigh" },
    TUPLE { PERSON# PERSON#(364), NAME "John Waters" },
    TUPLE { PERSON# PERSON#(406), NAME "Patricia Arquette" },
    TUPLE { PERSON# PERSON#(503), NAME "John Malkovich" },
    TUPLE { PERSON# PERSON#(615), NAME "Lars von Trier" },
    TUPLE { PERSON# PERSON#(639), NAME "Bryan Singer" },
    TUPLE { PERSON# PERSON#(745), NAME "Udo Kier" },
    TUPLE { PERSON# PERSON#(793), NAME "Jude Law" },
    TUPLE { PERSON# PERSON#(1070), NAME "David O. Russell" },
    TUPLE { PERSON# PERSON#(1485), NAME "Spike Jonze" },
    TUPLE { PERSON# PERSON#(1641), NAME "Iggy Pop" },
    TUPLE { PERSON# PERSON#(2020), NAME "Andrew Niccol" },
    TUPLE { PERSON# PERSON#(2259), NAME "Dario Argento" },
    TUPLE { PERSON# PERSON#(2557), NAME "Rupert Wainwright" },
    TUPLE { PERSON# PERSON#(2861), NAME "Stephen Norrington" },
    TUPLE { PERSON# PERSON#(3578), NAME "Traci Lords" }
};
```

```
CASTING := RELATION {
    TUPLE { MOVIE# MOVIE#(6), ACTOR# PERSON#(308), ORD 2 },
    TUPLE { MOVIE# MOVIE#(6), ACTOR# PERSON#(302), ORD 3 },
    TUPLE { MOVIE# MOVIE#(70), ACTOR# PERSON#(282), ORD 2 },
    TUPLE { MOVIE# MOVIE#(70), ACTOR# PERSON#(503), ORD 14 },
    TUPLE { MOVIE# MOVIE#(107), ACTOR# PERSON#(9), ORD 1 },
    TUPLE { MOVIE# MOVIE#(107), ACTOR# PERSON#(138), ORD 2 },
    TUPLE { MOVIE# MOVIE#(107), ACTOR# PERSON#(243), ORD 4 },
    TUPLE { MOVIE# MOVIE#(110), ACTOR# PERSON#(26), ORD 1 },
    TUPLE { MOVIE# MOVIE#(110), ACTOR# PERSON#(187), ORD 2 },
    TUPLE { MOVIE# MOVIE#(112), ACTOR# PERSON#(138), ORD 1 },
    TUPLE { MOVIE# MOVIE#(112), ACTOR# PERSON#(1485), ORD 4 },
    TUPLE { MOVIE# MOVIE#(151), ACTOR# PERSON#(243), ORD 2 },
    TUPLE { MOVIE# MOVIE#(151), ACTOR# PERSON#(793), ORD 3 },
    TUPLE { MOVIE# MOVIE#(213), ACTOR# PERSON#(745), ORD 6 },
    TUPLE { MOVIE# MOVIE#(213), ACTOR# PERSON#(3578), ORD 8 },
    TUPLE { MOVIE# MOVIE#(228), ACTOR# PERSON#(26), ORD 1 },
    TUPLE { MOVIE# MOVIE#(228), ACTOR# PERSON#(406), ORD 4 },
    TUPLE { MOVIE# MOVIE#(251), ACTOR# PERSON#(9), ORD 1 },
    TUPLE { MOVIE# MOVIE#(251), ACTOR# PERSON#(308), ORD 2 },
    TUPLE { MOVIE# MOVIE#(251), ACTOR# PERSON#(745), ORD 10 },
    TUPLE { MOVIE# MOVIE#(281), ACTOR# PERSON#(243), ORD 7 },
    TUPLE { MOVIE# MOVIE#(281), ACTOR# PERSON#(503), ORD 2 },
    TUPLE { MOVIE# MOVIE#(373), ACTOR# PERSON#(26), ORD 1 },
    TUPLE { MOVIE# MOVIE#(373), ACTOR# PERSON#(187), ORD 6 },
    TUPLE { MOVIE# MOVIE#(373), ACTOR# PERSON#(282), ORD 8 },
    TUPLE { MOVIE# MOVIE#(373), ACTOR# PERSON#(302), ORD 2 },
    TUPLE { MOVIE# MOVIE#(432), ACTOR# PERSON#(308), ORD 2 },
    TUPLE { MOVIE# MOVIE#(432), ACTOR# PERSON#(406), ORD 1 },
    TUPLE { MOVIE# MOVIE#(433), ACTOR# PERSON#(350), ORD 1 },
    TUPLE { MOVIE# MOVIE#(433), ACTOR# PERSON#(793), ORD 2 },
    TUPLE { MOVIE# MOVIE#(573), ACTOR# PERSON#(26), ORD 1 },
    TUPLE { MOVIE# MOVIE#(573), ACTOR# PERSON#(308), ORD 12 },
    TUPLE { MOVIE# MOVIE#(573), ACTOR# PERSON#(1641), ORD 6 },
    TUPLE { MOVIE# MOVIE#(1468), ACTOR# PERSON#(745), ORD 3 },
    TUPLE { MOVIE# MOVIE#(1512), ACTOR# PERSON#(745), ORD 9 },
    TUPLE { MOVIE# MOVIE#(1539), ACTOR# PERSON#(26), ORD 1 },
    TUPLE { MOVIE# MOVIE#(1539), ACTOR# PERSON#(1641), ORD 5 },
    TUPLE { MOVIE# MOVIE#(1539), ACTOR# PERSON#(3578), ORD 7 }
};
```

Database design

- Adding foreign keys

```
CONSTRAINT MOVIE_FKEY_DIRECTOR
  MOVIE {DIRECTOR#} RENAME {DIRECTOR# AS PERSON#} <= PERSON {PERSON#};
```

  - Take care about renaming!
  - If type names do not match, you must rename either one of the type names to match them
  - ITU convention: rename the more specific type to more generic type.

Database design

- Add other foreign keys

```
CONSTRAINT CASTING_FKEY_MOVIE
  CASTING {MOVIE#} <= MOVIE {MOVIE#};
```

```
CONSTRAINT CASTING_FKEY_ACTOR
  CASTING {ACTOR#} RENAME {ACTOR# AS PERSON#} <= PERSON {PERSON#};
```

Database design

- Foreign key constraints cannot be violated

```
DELETE PERSON
  WHERE (PERSON# = PERSON#(639));
```

  - 639th person exists

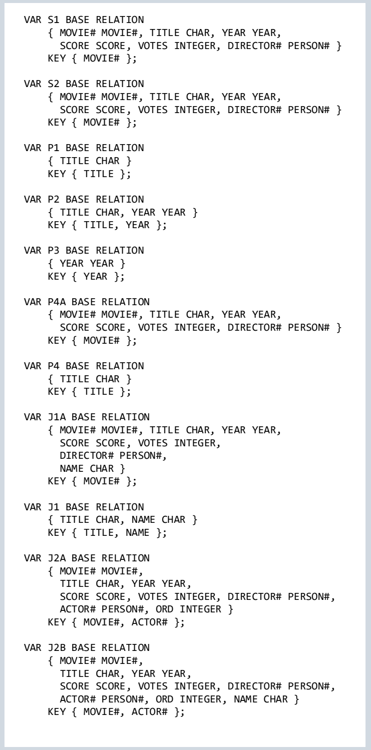
```
CASTING := RELATION{
  TUPLE{
    MOVIE# MOVIE#(13),
    ACTOR# PERSON#(666),
    ORD 1
  }
};
```

```
CASTING := RELATION{
  TUPLE{
    MOVIE# MOVIE#(70),
    ACTOR# PERSON#(666),
    ORD 1
  }
};
```

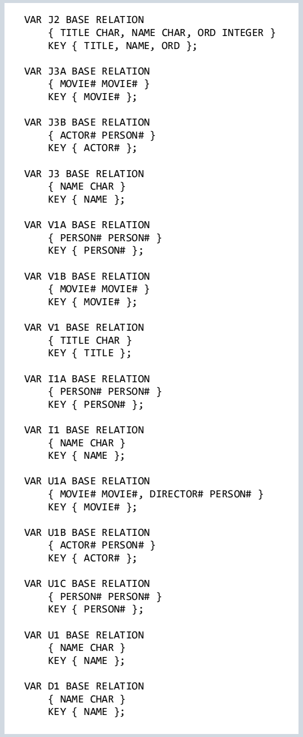
  - There is neither 13th movie nor 666th person exist

Database design

- The database is ready
  - You may want to define variables in bulk for relational algebra



```
VAR S1 BASE RELATION
    { MOVIE# MOVIE#, TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER, DIRECTOR# PERSON# }
    KEY { MOVIE# };

VAR S2 BASE RELATION
    { MOVIE# MOVIE#, TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER, DIRECTOR# PERSON# }
    KEY { MOVIE# };

VAR P1 BASE RELATION
    { TITLE CHAR }
    KEY { TITLE };

VAR P2 BASE RELATION
    { TITLE CHAR, YEAR YEAR }
    KEY { TITLE, YEAR };

VAR P3 BASE RELATION
    { YEAR YEAR }
    KEY { YEAR };

VAR P4A BASE RELATION
    { MOVIE# MOVIE#, TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER, DIRECTOR# PERSON# }
    KEY { MOVIE# };

VAR P4 BASE RELATION
    { TITLE CHAR }
    KEY { TITLE };

VAR J1A BASE RELATION
    { MOVIE# MOVIE#, TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER,
      DIRECTOR# PERSON#,
      NAME CHAR }
    KEY { MOVIE# };

VAR J1 BASE RELATION
    { TITLE CHAR, NAME CHAR }
    KEY { TITLE, NAME };

VAR J2A BASE RELATION
    { MOVIE# MOVIE#,
      TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER, DIRECTOR# PERSON#,
      ACTOR# PERSON#, ORD INTEGER }
    KEY { MOVIE#, ACTOR# };

VAR J2B BASE RELATION
    { MOVIE# MOVIE#,
      TITLE CHAR, YEAR YEAR,
      SCORE SCORE, VOTES INTEGER, DIRECTOR# PERSON#,
      ACTOR# PERSON#, ORD INTEGER, NAME CHAR }
    KEY { MOVIE#, ACTOR# };
```



```
VAR J2 BASE RELATION
    { TITLE CHAR, NAME CHAR, ORD INTEGER }
    KEY { TITLE, NAME, ORD };

VAR J3A BASE RELATION
    { MOVIE# MOVIE# }
    KEY { MOVIE# };

VAR J3B BASE RELATION
    { ACTOR# PERSON# }
    KEY { ACTOR# };

VAR J3 BASE RELATION
    { NAME CHAR }
    KEY { NAME };

VAR V1A BASE RELATION
    { PERSON# PERSON# }
    KEY { PERSON# };

VAR V1B BASE RELATION
    { MOVIE# MOVIE# }
    KEY { MOVIE# };

VAR V1 BASE RELATION
    { TITLE CHAR }
    KEY { TITLE };

VAR I1A BASE RELATION
    { PERSON# PERSON# }
    KEY { PERSON# };

VAR I1 BASE RELATION
    { NAME CHAR }
    KEY { NAME };

VAR U1A BASE RELATION
    { MOVIE# MOVIE#, DIRECTOR# PERSON# }
    KEY { MOVIE# };

VAR U1B BASE RELATION
    { ACTOR# PERSON# }
    KEY { ACTOR# };

VAR U1C BASE RELATION
    { PERSON# PERSON# }
    KEY { PERSON# };

VAR U1 BASE RELATION
    { NAME CHAR }
    KEY { NAME };

VAR D1 BASE RELATION
    { NAME CHAR }
    KEY { NAME };
```

Relational algebra

- Write an expression

```
MOVIE WHERE (VOTES > 10000)
```

- Or assign the result to a predefined variable by a statement

```
S1 := MOVIE WHERE (VOTES > 10000);
```

```
S1
```

```
VAR S1 BASE RELATION
  {MOVIE# MOVIE#,
   TITLE CHAR,
   YEAR YEAR,
   SCORE SCORE,
   VOTES INTEGER,
   DIRECTOR# PERSON#}
KEY {MOVIE#};
```

Relational algebra

```
P4A := MOVIE WHERE ( (VOTES > 5000) AND (SCORE > SCORE(7.0)) );
```

```
P4A
```

```
P4A {TITLE}
```

Relational algebra

- Movies & Directors

```
J1A := MOVIE JOIN (PERSON RENAME { PERSON# AS DIRECTOR# });
```

```
J1A
```

```
J1A {TITLE, NAME}
```

Relational algebra

- Johnny Depp & Friends

```
J3A := (((PERSON RENAME { PERSON# AS ACTOR# }) JOIN CASTING)
  WHERE (NAME = "Johnny Depp")) { MOVIE# };
```

```
J3B := (J3A JOIN CASTING) { ACTOR# };
```

```
((J3B RENAME { ACTOR# AS PERSON# }) JOIN PERSON) {NAME}
```