Database Systems

Relational Model

H. Turgut Uyar Şule Öğüdücü

2002-2012

License

Topics

Data Models

- previous models:
  - inverted list
  - hierarchical
  - network
- relational model:
  - Dr. E. F. Codd, 1970
- later models:
  - object
  - object / relational

Relational Model

- data is modelled as relations:
  $\alpha \subseteq A \times B \times C \times \ldots$
- relations are assigned to relation variables (relvar)
- each element of a relation is a tuple
- each piece of data of an element is an attribute
- relations are represented using tables
  - the user should *perceive* all data as tables
  - relation → table, tuple → row, attribute → column

Relation Example

Example (movie relation)

MOVIE

TITLE	YEAR	DIRECTOR	SCORE	VOTES
Usual Suspects	1995	Bryan Singer	8.7	3502
Suspiria	1977	Dario Argento	7.1	1004
Being John Malkovich	1999	Spike Jonze	8.3	13809
...	...	...	...	...

- the name of the relation variable is `MOVIE`
- `(Usual Suspects, 1995, Bryan Singer, 8.7, 3502)` is a tuple of the movie relation
- `YEAR` is an attribute of the movie relation

Relation Structure

relation header

- the set of attributes of the relation
- affected by data definition language statements

relation body

- the set of tuples in the relation
- affected by data manipulation language statements

Relation Predicate

Definition
relation predicate:
the sentence expressing the "meaning" of the relation

- each tuple is either *True* or *False* according to the predicate

Relation Predicate Example

Example (movie relation predicate)

- The movie with the title `TITLE` was filmed in the year `YEAR`, by the director `DIRECTOR`; the average of `VOTES` votes is `SCORE`.
- the tuple `(Suspiria, 1977, Dario Argento, 1004, 7.1)` is True
- the tuple `(Suspiria, 1978, Dario Argento, 1004, 7.1)` is False

Tuple Order

- the order of tuples is insignificant

Example

- the following two relations are equivalent:

TITLE	...
Usual Suspects	...
Suspiria	...
Being John Malkovich	...

TITLE	...
Suspiria	...
Being John Malkovich	...
Usual Suspects	...

Attribute Order

- the order of attributes is insignificant

Example

- the following two relations are equivalent:

TITLE	YEAR	...
Usual Suspects	1995	...
Suspiria	1977	...
Being John Malkovich	1999	...

YEAR	TITLE	...
1995	Usual Suspects	...
1977	Suspiria	...
1999	Being John Malkovich	...

Attribute Values

- attribute values must be scalar
  - arrays, lists, records etc. are not allowed

Example (multiple directors)

TITLE	...	DIRECTORS	...
...	...	...	...
Matrix	...	~~Andy Wachowski, Lana Wachowski~~	...
...	...	...	...

Null Value

- the value of the attribute is not known for this tuple

Example

- the director of the movie is not known

- this tuple does not have a value for this attribute

Example

- nobody voted for the movie, therefore there is no SCORE

Default Value

- default values can be used instead of null values
  - it may not be one of the valid values for the attribute

Example

- if SCORE values are between 1.0 and 10.0, the default value can be chosen as 0.0

Duplicate Tuples

- there can not be duplicate tuples in a relation
  - each tuple must be uniquely identifiable

Example

TITLE	YEAR	DIRECTOR	SCORE	VOTES
Usual Suspects	1995	Bryan Singer	8.7	3502
Suspiria	1977	Dario Argento	7.1	1004
Being John Malkovich	1999	Spike Jonze	8.3	13809
...	...	...	...	...
Suspiria	1977	Dario Argento	7.1	1004
...	...	...	...	...

Keys

- let B be the set of all attributes of the relation, and let $A \subseteq B$
- in order for A to be a candidate key, the following conditions must hold:
  - uniqueness: no two tuples have the same values for all attributes in A
  - irreducibility: no subset of A satisfies the uniqueness property
- every relation has at least one candidate key

Candidate Key Example

Example (candidate keys for movie relation)

- {TITLE}
- {TITLE, YEAR}
- {TITLE, DIRECTOR}
- {TITLE, YEAR, DIRECTOR}

Surrogate Keys

- if a natural key can not be found a surrogate key can be defined
- identity attribute
  - its value does not matter
  - it can be generated by the system

Surrogate Key Example

Example

MOVIE#	TITLE	YEAR	DIRECTOR	SCORE	VOTES
...	...	...	...	...	...
6	Usual Suspects	1995	Bryan Singer	...	...
1512	Suspiria	1977	Dario Argento	...	...
70	Being John Malkovich	1999	Spike Jonze	...	...
...	...	...	...	...	...

- `{MOVIE#}` is a candidate key
- `{MOVIE#, TITLE}` is not a candidate key

Primary Key

- if a relation has more than one candidate key:
  - one of them is selected as the primary key
  - the others are alternate keys
- every relation must have a primary key
- any attribute that is part of the primary key can not be empty in any tuple

Primary Key Example

- the names of the attributes in the primary key are underlined

Example

MOVIE#	TITLE	YEAR	DIRECTOR	SCORE	VOTES
...	...	...	...	...	...
6	Usual Suspects	1995	Bryan Singer	...	...
1512	Suspiria	1977	Dario Argento	...	...
70	Being John Malkovich	1999	Spike Jonze	...	...
...	...	...	...	...	...

Data Types

- all values for the same attribute have to be selected from the same domain
  - comparison only makes sense between values chosen from the same domain
- in practice, data types are used instead

Domain Example

Example

- `TITLE` from the titles domain, `YEAR` from the years domain, `DIRECTOR` from the directors domain, ...
- if data types are used:
  `TITLE` string, `YEAR` integer, `DIRECTOR` string, ...
  - assigning the value `"Usual Suspects"` to the attribute `DIRECTOR` is correct in terms of data types but incorrect according to predicate
  - `YEAR` and `VOTES` are integers but it does not make sense to compare them

Tutorial D Data Types

- `INTEGER`
- `RATIONAL`
- `BOOL`
- `CHAR`

Defining Types

Statement

```
TYPE type_name POSSREP {
  field_name field_type
  [, ...]
};
```

Deleting Types

```
DROP TYPE type_name;
```


Type Definition Examples

Example

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

TYPE YEAR POSSREP { VALUE INTEGER };

TYPE SCORE POSSREP { VALUE RATIONAL };
```


Type Operations

- generating a value for a derived type:

  ```
  type_name(base_value [, ...])
  ```

- getting the value of a field: `THE_` operators

  ```
  THE_field_name(variable_name)
  ```

- type casting: `CAST_AS_` operators

  ```
  CAST_AS_target_type(value)
  ```

- renaming an attribute:

  ```
  RENAME (attribute_name AS new_name)
  ```


Type Operation Examples

Example

- generating a SCORE value:

  ```
  SCORE(8.7)
  ```

- getting the VALUE field of a SCORE variable:

  ```
  THE_VALUE(SCORE)
  ```

- casting an integer VOTES value to a RATIONAL:

  ```
  CAST_AS_RATIONAL(VOTES)
  ```

- renaming the MOVIE# attribute:

  ```
  RENAME (MOVIE# AS MOVIENO)
  ```


Value Constraints

Statement

```
TYPE type_name POSSREP {
  field_name field_type
  [, ...]
  CONSTRAINT condition
};
```


Value Constraint Example

Example

- SCORE values must be between 1.0 and 10.0

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

Defining Relations

Statement

```
RELATION
  { attribute_name attribute_type
    [, ...] }
  KEY { attribute_name [, ...] }
```


Relation Definition Example

Example

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


Creating Base Relation Variables

Statement

```
VAR relvar_name BASE RELATION
  { ... }
  KEY { ... };
```

Deleting Variables

```
DROP VAR relvar_name;
```


Base Relation Variable Creation Example

Example

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


Tuple and Relation Generation

Tuple Generation

```
TUPLE {
  attribute_name attribute_value
  [, ...]
}
```

Relation Generation

```
RELATION {
  TUPLE { ... }
  [, ...]
}
```


Relation Variable Assignment

- assigning relations to relation variables:

```
relvar_name := RELATION { ... };
```

Relation Variable Assignment Example

Example

```
MOVIE := RELATION {
  TUPLE { MOVIE# MOVIE#(6),
    TITLE "Usual Suspects",
    YEAR YEAR(1995), DIRECTOR "Bryan Singer",
    SCORE SCORE(8.7), VOTES 35027 },
  TUPLE { MOVIE# MOVIE#(70),
    TITLE "Being John Malkovich",
    YEAR YEAR(1999), DIRECTOR "Spike Jonze",
    SCORE SCORE(8.3), VOTES 13809 }
};
```


Inserting Tuples

Statement

```
INSERT relvar_name RELATION {
  TUPLE { ... }
  [, ...]
};
```


Tuple Insertion Example

Example

```
INSERT MOVIE RELATION {
  TUPLE { MOVIE# MOVIE#(6),
    TITLE "Suspiria",
    YEAR YEAR(1977),
    DIRECTOR "Dario Argento",
    SCORE SCORE(7.1), VOTES 1004 }
};
```


Deleting Tuples

Statement

```
DELETE relvar_name
  [ WHERE condition ];
```

- if no condition is specified, all tuples will be deleted

Tuple Deletion Example

Example

- delete movies with scores less than 3.0 and votes more than 4

```
DELETE MOVIE
  WHERE ((SCORE < SCORE(3.0))
     AND (VOTES > 4));
```


Updating Tuples

Statement

```
UPDATE relvar_name
  [ WHERE condition ]
  ( attribute_name := attribute_value
    [, ...] );
```

- if no condition is specified, all tuples will be updated

Tuple Update Example

Example

- register a new vote (9) for the movie "Suspiria"

```
UPDATE MOVIE
  WHERE (TITLE = "Suspiria") (
    SCORE := SCORE(
      (THE_VALUE(SCORE)
          * CAST_AS_RATIONAL(VOTES)
          + CAST_AS_RATIONAL(9))
        / CAST_AS_RATIONAL(VOTES + 1)
    ),
    VOTES := VOTES + 1
  );
```


Scalar Values

- in order to meet the scalar value requirement, tuples may have to be repeated

Example (how to store actor data?)

MOVIE

MOVIE#	TITLE	...	ACTORS
6	Usual Suspects	...	Gabriel Byrne
...	...	...	...
70	Being John Malkovich	...	~~Cameron Diaz, John Malkovich~~
...	...	...	...

MOVIE

MOVIE#	TITLE	...	ACTOR
6	Usual Suspects	...	Gabriel Byrne
...	...	...	...
70	Being John Malkovich	...	Cameron Diaz
70	Being John Malkovich	...	John Malkovich
...	...	...	...

Scalar Value Example

Example (movies and actors)

MOVIE

MOVIE#	TITLE	...
6	Usual Suspects	...
1512	Suspiria	...
70	Being John Malkovich	...
...	...	...

ACTOR

ACTOR#	NAME
308	Gabriel Byrne
282	Cameron Diaz
503	John Malkovich
...	...

CASTING

MOVIE#	ACTOR#	ORD
6	308	2
70	282	2
70	503	14
...	...	...

Scalar Value Example

Example (how to store director data?)

MOVIE

MOVIE#	TITLE	...	DIRECTOR#
6	Usual Suspects	...	639
1512	Suspiria	...	2259
70	Being John Malkovich	...	1485
...	...	...	...

PERSON

PERSON#	NAME
308	Gabriel Byrne
1485	Spike Jonze
639	Bryan Singer
282	Cameron Diaz
2259	Dario Argento
503	John Malkovich
...	...

CASTING

MOVIE#	ACTOR#	ORD
6	308	2
70	282	2
70	503	14
...	...	...

Foreign Keys

Definition

foreign key: an attribute of a relation being the candidate key of another relation

Foreign Key Examples

Example (the DIRECTOR# attribute of the MOVIE relation variable)

MOVIE

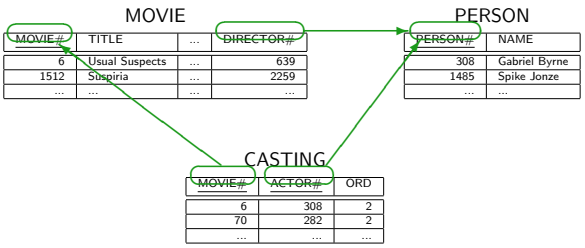
MOVIE#	TITLE	...	~~DIRECTOR#~~
6	Usual Suspects	...	639
1512	Suspiria	...	2259
70	Being John Malkovich	...	1485
...	...	...	...

PERSON

~~PERSON#~~	NAME
308	Gabriel Byrne
1485	Spike Jonze
639	Bryan Singer
282	Cameron Diaz
2259	Dario Argento
503	John Malkovich
...	...

Foreign Key Example

Example (foreign keys in the movie database)



Referential Integrity

Definition

referential integrity:
all values of a foreign key attribute must be present
in the corresponding attribute of the referenced relation

- prevent a request if it conflicts with referential integrity

Referential Integrity Example

Example

MOVIE

MOVIE#	TITLE	...	DIRECTOR#
...	...	...	...
1512	Suspiria	...	2259
...	...	...	...

PERSON

PERSON#	NAME
...	...
2259	Dario Argento
...	...

- delete the tuple `(2259, Dario Argento)`
- update the tuple `(2259, Dario Argento)` as `(2871,Dario Argento)`

Defining Foreign Keys

Statement

```
CONSTRAINT constraint_name
  referencing_relvar_name
      { attribute_name }
    <= referenced_relvar_name
        { attribute_name };
```

- attribute names have to match in both relations
  - otherwise, attributes should be renamed

Foreign Key Definition Examples

Example (the `MOVIE#` attribute of the `CASTING` relation variable)

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


Foreign Key Definition Examples

Example (the `DIRECTOR#` attribute of the `MOVIE` relation variable)

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

Creating the Example Database

Example (types)

```
TYPE MOVIE# POSSREP { VALUE INTEGER };
TYPE YEAR POSSREP { VALUE INTEGER };
TYPE SCORE POSSREP { VALUE RATIONAL
  CONSTRAINT (VALUE >= 1.0)
          AND (VALUE <= 10.0) };
TYPE PERSON# POSSREP { VALUE INTEGER };
```


Creating the Example Database

Example (the MOVIE relation variable)

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


Creating the Example Database

Example (the PERSON relation variable)

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


Creating the Example Database

Example (the CASTING relation variable)

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


Creating the Example Database

Example (the foreign keys of the MOVIE relation variable)

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


Creating the Example Database

Example (the foreign keys of the CASTING relation variable)

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

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

Data Types

- INTEGER
  - SMALLINT
- NUMERIC (precision, scale)
  - precision: total number of digits
  - scale: number of digits after the decimal point
  - same as: DECIMAL (precision, scale)
- FLOAT (p)
  - p: lowest acceptable precision
- BOOLEAN

String Data Types

- CHARACTER [VARYING] (n)
  - in CHARACTER (n), if the string is shorter than n characters it will be padded with spaces
- abbreviations:
  - CHAR (n) instead of CHARACTER (n)
  - VARCHAR (n) instead of CHARACTER VARYING (n)

Date / Time Data Types

- DATE
  - value example: 2005-09-26
- TIME
  - value example: 11:59:22.078717
- TIMESTAMP
  - value example: 2005-09-26 11:59:22.078717
- INTERVAL
  - value example: 3 days

Large Object Data Types

- arbitrary length objects
- can not be used in queries
- text: CHARACTER LARGE OBJECT (n)
  - CLOB
- binary: BINARY LARGE OBJECT (n)
  - BLOB
  - picture, audio, etc.

Creating Domains

Statement

```
CREATE DOMAIN domain_name [ AS ] base_type
  [ DEFAULT default_value ]
  [ { CHECK ( condition ) } [, ...] ]
```

Deleting Domains

```
DROP DOMAIN domain_name [, ...]
```


Domain Example

Example (a domain for valid SCORE values)

```
CREATE DOMAIN SCORES AS FLOAT
  CHECK ((VALUE >= 1.0) AND (VALUE <= 10.0))
```

Creating Tables

Statement

```
CREATE TABLE table_name (
  { column_name data_type }
  [, ... ]
)
```

Deleting Tables

```
DROP TABLE table_name [, ... ]
```


Null and Default Values

Statement

```
CREATE TABLE table_name (
  { column_name data_type
              [ NULL | NOT NULL ]
              [ DEFAULT default_value ] }
  [, ... ]
)
```

- `NULL`: the attribute is allowed to be empty (default)
- `NOT NULL`: the attribute is not allowed to be empty

Defining Primary Keys

Statement

```
CREATE TABLE table_name (
  { column_name data_type
              [ NULL | NOT NULL ]
              [ DEFAULT default_value ] }
  [, ... ]
  [ PRIMARY KEY ( column_name [, ...] ) ]
)
```

- if the primary key consists of a single column, it can be specified when defining the column:

  ```
  column_name data_type PRIMARY KEY
  ```


Table Creation Example

Example

```
CREATE TABLE MOVIE (
  ID INTEGER,
  TITLE VARCHAR(80) NOT NULL,
  YR NUMERIC(4),
  DIRECTOR VARCHAR(40),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  PRIMARY KEY (ID)
)
```


Table Creation Example

Example

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  ...
  VOTES INTEGER DEFAULT 0
)
```


Automatically Incremented Values

- product-specific definitions
  - PostgreSQL: `SERIAL` data type
  - MySQL: `AUTO_INCREMENT` property

Automatically Incremented Value Examples

Example (PostgreSQL)

```
CREATE TABLE MOVIE (
  ID SERIAL PRIMARY KEY,
  ...
)
```

Example (MySQL)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY AUTO_INCREMENT,
  ...
)
```


Value Constraints

Statement

```
CREATE TABLE table_name (
  ...
  [ { CHECK ( condition ) }
    [, ...] ]
  ...
)
```


Value Constraint Example

Example

- SCORE values must be between 1.0 and 10.0

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  ...,
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  CHECK ((SCORE >= 1.0) AND (SCORE <= 10.0))
)
```


Uniqueness Definition

Statement

```
CREATE TABLE table_name (
  ...
  [ { UNIQUE ( column_name [, ...] ) }
    [, ...] ]
  ...
)
```

- if the uniqueness constraint consists of a single column, it can be specified directly when defining the column:

  ```
  column_name data_type UNIQUE
  ```


Uniqueness Definition Example

Example

- {TITLE} and {DIRECTOR, YR} are unique

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  TITLE VARCHAR(80) UNIQUE NOT NULL,
  YR NUMERIC(4),
  DIRECTOR VARCHAR(40),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  UNIQUE (DIRECTOR, YR)
)
```


Renaming Tables

Statement

```
ALTER TABLE table_name
  RENAME TO new_name
```

Example

```
ALTER TABLE MOVIE
  RENAME TO FILM
```

Adding Columns

Statement

```
ALTER TABLE table_name
  ADD [ COLUMN ] column_name data_type
                 [ NULL | NOT NULL ]
                 [ DEFAULT default_value ]
```

Example

```
ALTER TABLE MOVIE
  ADD COLUMN RUNTIME INTEGER
```


Deleting Columns

Statement

```
ALTER TABLE table_name
  DROP [ COLUMN ] column_name
```

Example

```
ALTER TABLE MOVIE
  DROP COLUMN RUNTIME
```


Renaming Columns

Statement

```
ALTER TABLE table_name
  RENAME [ COLUMN ] column_name TO new_name
```

Example

```
ALTER TABLE MOVIE
  RENAME COLUMN TITLE TO NAME
```


Column Default Value

Setting Column Default Value

```
ALTER TABLE table_name
  ALTER [ COLUMN ] column_name
  SET DEFAULT default_value
```

Removing Column Default Value

```
ALTER TABLE table_name
  ALTER [ COLUMN ] column_name
  DROP DEFAULT
```


Adding Constraints

Statement

```
ALTER TABLE table_name
  ADD [ CONSTRAINT constraint_name ]
    constraint_definition
```

- what about existing tuples?

Removing Constraints

```
ALTER TABLE table_name
  DROP [ CONSTRAINT ] constraint_name
```


Constraint Addition Example

Example

- YR values can not be less than 1888

```
ALTER TABLE MOVIE
  ADD CONSTRAINT MINIMUM_YEAR
    CHECK (YR >= 1888)
```

- drop the minimum year constraint

```
ALTER TABLE MOVIE
  DROP CONSTRAINT MINIMUM_YEAR
```

Inserting Rows

Statement

```
INSERT INTO table_name
  [ ( column_name [, ...] ) ]
  VALUES ( column_value [, ...] )
```

- the order of values must match the order of columns
- if the column names are not specified, values must be written in the order used at table creation
- unspecified attributes will take their default values
- automatically generated columns should not be specified

Row Insertion Examples

Example

```
INSERT INTO MOVIE VALUES (
  6,
  'Usual Suspects',
  1995,
  'Bryan Singer',
  8.7,
  35027
)
```


Row Insertion Examples

Example

```
INSERT INTO MOVIE (YR, TITLE) VALUES (
  1995,
  'Usual Suspects'
)
```

- the ID value will be automatically generated

Deleting Rows

Statement

```
DELETE FROM table_name
  [ WHERE condition ]
```


Row Deletion Example

Example

- delete movies with scores less than 3.0 and votes more than 4

```
DELETE FROM MOVIE
  WHERE ((SCORE < 3.0) AND (VOTES > 4))
```


Updating Rows

Statement

```
UPDATE table_name
  SET { column_name = column_value } [, ...]
  [ WHERE condition ]
```

Row Update Example

Example

- register a new vote (9) for the movie "Suspiria"

```
UPDATE MOVIE
  SET SCORE = (SCORE * VOTES + 9)
                / (VOTES + 1),
      VOTES = VOTES + 1
  WHERE (TITLE = 'Suspiria')
```


Defining Foreign Keys

Statement

```
CREATE TABLE table_name (
  ...
  [ { FOREIGN KEY ( column_name [, ...] )
        REFERENCES table_name
          [ ( column_name [, ...] ) ]
        [ ON DELETE option ]
        [ ON UPDATE option ] } [, ...] ]
  ...
)
```


Referential Integrity Options

- do not allow the operation: `RESTRICT`, `NO_ACTION`
- reflect the change to affected tuples: `CASCADE`
- assign null value: `SET NULL`
  - if null values are allowed
- assign default value: `SET DEFAULT`

Referential Integrity Examples

Example (restrict when deleting)

- if the `ID` attribute of the director to be deleted from the `PERSON` relation is present among the current values of the `DIRECTORID` attribute of the `MOVIE` relation, do not allow deleting

Example (cascade when updating)

- when the `ID` attribute value of a director has been updated in the `PERSON` relation, also update the `DIRECTORID` attribute of the corresponding tuples in the `MOVIE` relation

Referential Integrity Example

Example (cascade when deleting)

- delete a person from the `PERSON` relation
- delete all tuples from the `CASTING` relation where the `ACTORID` attribute has the same value as the `ID` attribute of the deleted person
- delete all tuples from the `MOVIE` relation where the `DIRECTORID` attribute has the same value as the `ID` attribute of the deleted person
- for each deleted `MOVIE` tuple in the previous step, delete all tuples from the `CASTING` relation where the `MOVIEID` attribute has the same value as the `ID` attribute of the deleted movie

Foreign Key Definition Example

Example (the `DIRECTORID` foreign key in the `MOVIE` table)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  ...
  DIRECTORID INTEGER,
  FOREIGN KEY DIRECTORID
    REFERENCES PERSON (ID)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
)
```

Defining Foreign Keys

- if the foreign key corresponds to the primary key in the referenced table, it does not have to be specified in the REFERENCES part
- if the foreign key consists of only one column, it can be specified when defining the column:

```
column_name data_type
  REFERENCES table_name [ ( column_name ) ]
```


Creating the Example Database

Example (creating the MOVIE table)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  TITLE VARCHAR(80) NOT NULL,
  YR NUMERIC(4),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  DIRECTORID INTEGER REFERENCES PERSON
)
```


Creating the Example Database

Example (creating the PERSON table)

```
CREATE TABLE PERSON (
  ID INTEGER PRIMARY KEY,
  NAME VARCHAR(40) UNIQUE NOT NULL
)
```


Creating the Example Database

Example (creating the CASTING table)

```
CREATE TABLE CASTING (
  MOVIEID INTEGER REFERENCES MOVIE,
  ACTORID INTEGER REFERENCES PERSON,
  ORD INTEGER,
  PRIMARY KEY (MOVIEID, ACTORID)
)
```


References

Required Reading: Date

- Chapter 3: An Introduction to Relational Databases
  - 3.2. An Informal Look at the Relational Model
  - 3.3. Relations and Relvars
- Chapter 6: Relations
- Chapter 9: Integrity
  - 9.10. Keys
  - 9.12. SQL Facilities