Database Systems

Relational Model

H. Turgut Uyar Şule Öğüdücü

2002-2010

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

Relation Example

Example (movie relation)

Table: MOVIE

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

- `(Usual Suspects,1995,Bryan Singer,UK,8.7,3502)` is a tuple of the MOVIE relation
- `COUNTRY` is an attribute of the MOVIE relation

Relation Structure

relation header

- the set of attributes making up the relation
- specified when the relation is created
- affected by the data definition language statements

relation body

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

Relation Predicate

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

- each tuple takes the value *True* or *False* according to this predicate

Relation Predicate Example

Example (movie relation predicate)

- the movie titled as `TITLE` was filmed in the year `YR`, by the director `DIRECTOR`, in the country `COUNTRY`; the average of `VOTES` votes is `SCORE`.
- the tuple `(Suspiria,1977,Dario Argento,IT,1004,7.1)` is True
- the tuple `(Suspiria,1877,Dario Argento,IT,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	YR	...
Usual Suspects	1995	...
Suspiria	1977	...
Being John Malkovich	1999	...

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

Identical Tuples

- there can not be identical tuples in a relation
  - each tuple has to be uniquely identifiable

Example

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

Domain

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

Domain Example

Example

- `TITLE` from the titles domain, `YR` from the years domain, `COUNTRY` from the countries domain, ...
- if data types are used:
  `TITLE` string, `YR` integer, `COUNTRY` string, ...
  - assigning the value `Woody Allen` to the attribute `COUNTRY` is correct in terms of data types, but wrong according to predicate
  - `YR` and `VOTES` are integers but it does not make sense to compare them

Null Value

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

Example

- the director of the movie `Blade` is not known

- this tuple does not have a value for the attribute

Example

- no votes for the movie `Star Wars` yet, therefore `SCORE` is empty

Default Value

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

Example

- the default value for the `SCORE` attribute can be chosen as `-1`

Data Definition Language

- creating and deleting relations
- changing the relation header
  - changing the relation name
  - adding attributes
  - changing the name of an attribute

SQL Data Types

- number
- string
- boolean
- date/time
- large objects

Basic 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 Types

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

Date/Time 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 Types

- arbitrary length
- can not be used in queries

- binary: **BINARY LARGE OBJECT** (n)
  - **BLOB**
  - picture, audio, etc.
- text: **CHARACTER LARGE OBJECT** (n)
  - **CLOB**

Creating Tables

Statement

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

Deleting Tables

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


Table Deletion Example

Example

```
CREATE TABLE MOVIE (
  TITLE VARCHAR(80),
  YR NUMERIC(4),
  DIRECTOR VARCHAR(40),
  COUNTRY CHAR(2),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0
)
```

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
                 [ DEFAULT default_value ]
```

Example

```
ALTER TABLE MOVIE
  ADD COLUMN LANGUAGE CHAR(2)
```


Deleting Columns

Statement

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

Example

```
ALTER TABLE MOVIE
  DROP COLUMN LANGUAGE
```


Renaming Columns

Statement

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

Example

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


Changing Column Default Value

Statement

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

Example

```
ALTER TABLE MOVIE
  ALTER COLUMN SCORE
  SET DEFAULT -1
```


Removing Column Default Value

Statement

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

Example

```
ALTER TABLE MOVIE
  ALTER COLUMN SCORE
  DROP DEFAULT
```

Data Manipulation Language

- adding tuples
- deleting tuples
- updating tuples

- operations are carried out on tuple sets
  - all tuples satisfying a condition

Adding Rows

Statement

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

- the order of values has to match the order of columns
- unspecified attributes will take default value
- if the column names are not specified, the column values have to given in the order used when creating the table

Row Adding Example

Example

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


Row Adding Example

Example

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


Deleting Rows

Statement

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

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

Row Deleting Example

Example (delete all movies with scores less than 3)

```
DELETE FROM MOVIE
  WHERE (SCORE < 3)
```

Row Deleting Example

Example (delete all movies with scores less than 3 and at least 5 votes)

```
DELETE FROM MOVIE
  WHERE ((SCORE < 3) AND (VOTES >= 5))
```


Updating Rows

Statement

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

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

Row Updating Example

Example (reset the scores and votes of all movies filmed before 1997 to zero)

```
UPDATE MOVIE
  SET SCORE = 0, VOTES = 0
  WHERE (YR < 1997)
```


Row Updating Example

Example (vote 9 for "Suspiria")

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


Integrity

- *problem*: preventing tuples that contradict the relation predicate
- rules are expressed using the data definition language
  - value constraints
  - keys

Value Constraint

Definition

value constraint:
values must satisfy a boolean expression

Example

- `YR` value can not be smaller than 1880
- `SCORE` values must be between 0 and 10
- `TITLE` values can not be empty

Specifying Value Constraints

Statement

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


Value Constraint Example

Example (YR values can not be smaller than 1880)

```
CREATE TABLE MOVIE (
  TITLE VARCHAR(80),
  YR NUMERIC(4),
  ...,
  VOTES INTEGER DEFAULT 0,
  CHECK (YR >= 1880)
)
```


Null Value Constraint

Expressing NULL condition

```
CREATE TABLE table_name (
  ...
  CHECK ( column_name IS { NULL | NOT NULL } )
  ...
)
```

- can be specified at column definition:
  column_name data_type [{ **NULL** | **NOT NULL** }]

Null Value Constraint Example

Example (TITLE attributes can not be empty)

```
CREATE TABLE MOVIE (
  TITLE VARCHAR(80),
  YR NUMERIC(4),
  ..,
  VOTES INTEGER DEFAULT 0,
  CHECK (TITLE IS NOT NULL)
)
```


Null Value Constraint Example

Example (TITLE attributes can not be empty)

```
CREATE TABLE MOVIE (
  TITLE VARCHAR(80) NOT NULL,
  YR NUMERIC(4),
  ...,
  VOTES INTEGER DEFAULT 0
)
```


Creating Domains

Statement

```
CREATE DOMAIN domain_name [ AS ] base_type
  [ DEFAULT default_value ]
  [ constraint_definition [, ...] ]
```

- the created domain can be used as a column type

Deleting Domains

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

Domain Example

Example (domain for year values)

```
CREATE DOMAIN YEARS AS NUMERIC(4)
  DEFAULT 2005
  CHECK (VALUE >= 1880)

CREATE TABLE MOVIE (
  TITLE VARCHAR(80),
  YR YEARS,
  ...
)
```


Managing 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 Management Example

Example (SCORE values can not be greater than 10)

```
ALTER TABLE MOVIE
  ADD CHECK (SCORE <= 10)
```


Constraint Management Examples

Example (SCORE values can not be smaller than 0)

```
ALTER TABLE MOVIE
  ADD CONSTRAINT SCORE_POSITIVE
      CHECK (SCORE >= 0)
```

Example (SCORE values can be smaller than 0)

```
ALTER TABLE MOVIE
  DROP CONSTRAINT SCORE_POSITIVE
```


Candidate Key

- let B be the 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 the attributes in A
  - irreducibility: no subset of A satisfies the uniqueness property
- every relation has at least one candidate key
- every candidate key is a constraint

Candidate Key Example

Example (candidate keys for movie relation)

- {TITLE}
- {TITLE,YR}
- {TITLE,DIRECTOR}
- {TITLE,YR,DIRECTOR}

Surrogate Keys

- if a *natural key* can not be selected a *surrogate key* can be defined
- identity attributes
  - can be generated by the system

Surrogate Key Example

Example

Table: MOVIE

ID	TITLE	YR	DIRECTOR	COUNTRY	SCORE	VOTES
...	...	...	...	...	...	...
6	Usual Suspects	1995	Bryan Singer	...	...	...
1512	Suspiria	1977	Dario Argento	...	...	...
70	Being John Malkovich	1999	Spike Jonze	...	...	...
...	...	...	...	...	...	...

- `{ID}` is a candidate key
- `{ID,TITLE}` is not a candidate key

Defining Candidate Keys

Statement

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

- if a candidate key consists of only one column, it can be specified when the column is defined:
  column_name data_type **UNIQUE**

Candidate Key Definition Example

Example (`ID` and `{TITLE,DIRECTOR}` are unique)

```
CREATE TABLE MOVIE (
  ID INTEGER,
  TITLE VARCHAR(80),
  ...
  VOTES INTEGER DEFAULT 0,
  CHECK (YR >= 1880),
  UNIQUE (ID),
  UNIQUE (TITLE, DIRECTOR)
)
```


Candidate Key Definition Example

Example (`ID` is unique and can not be empty)

```
CREATE TABLE MOVIE (
  ID INTEGER UNIQUE NOT NULL,
  TITLE VARCHAR(80),
  ...
  VOTES INTEGER DEFAULT 0,
  CHECK (YR >= 1880),
  UNIQUE (TITLE , DIRECTOR)
)
```


Automatic Identity Values

- product-specific definitions
  - PostgreSQL: `SERIAL` data type
  - MySQL: `AUTO_INCREMENT` property
- not specified when adding rows

Automatic Identity Value Example

Example (PostgreSQL)

```
CREATE TABLE MOVIE (
  ID SERIAL UNIQUE NOT NULL,
  TITLE VARCHAR(80),
  ...
)
```


Automatic Identity Value Example

Example (MySQL)

```
CREATE TABLE MOVIE (
  ID INTEGER UNIQUE NOT NULL AUTO_INCREMENT,
  TITLE VARCHAR(80),
  ...
)
```


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

Primary Key Example

Example

Table: MOVIE

ID	TITLE	YR	DIRECTOR	COUNTRY	SCORE	VOTES
...	...	...	...	...	...	...
6	Usual Suspects	1995	Bryan Singer	...	...	...
1512	Suspiria	1977	Dario Argento	...	...	...
70	Being John Malkovich	1999	Spike Jonze	...	...	...
...	...	...	...	...	...	...

Defining Primary Keys

Statement

```
CREATE TABLE table_name (
  ...
  [ PRIMARY KEY ( column_name [, ...] ) ]
  ...
)
```

- if the primary key consists of only one column, it can be specified when the column is defined:
  column_name data_type **PRIMARY KEY**

Primary Key Definition Example

Example (ID is the primary key)

```
CREATE TABLE MOVIE (
  ID INTEGER,
  TITLE VARCHAR(80),
  ...
  VOTES INTEGER DEFAULT 0,
  CHECK (YR >= 1880),
  UNIQUE (TITLE, DIRECTOR),
  PRIMARY KEY (ID)
)
```

Primary Key Definition Example

Example (ID is the primary key)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  TITLE VARCHAR(80),
  ...
  VOTES INTEGER DEFAULT 0,
  CHECK (YR >= 1880),
  UNIQUE (TITLE, DIRECTOR)
)
```


Scalar Values

- attribute values must be scalar
  - no arrays, lists, records etc. are allowed
- values have to be duplicated in order to meet this requirement
  - duplication causes problems

Scalar Value Example

Example (how to store actor data?)

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

ID	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)

Table: MOVIE

ID	TITLE	...
6	Usual Suspects	...
1512	Suspiria	...
70	Being John Malkovich	...
...	...	...

Table: ACTOR

ID	NAME
308	Gabriel Byrne
282	Cameron Diaz
503	John Malkovich
...	...

Table: CASTING

MOVIEID	ACTORID	ORD
6	308	2
70	282	2
70	503	14
...	...	...

Scalar Value Example

Example (how to store director data?)

- separate relation or together with actors?

Table: MOVIE

ID	TITLE	...	DIRECTORID
6	Usual Suspects	...	639
1512	Suspiria	...	2259
70	Being John Malkovich	...	1485
...	...	...	...

Table: PERSON

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

Creating the Example Tables

Example (creating the MOVIE table)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  TITLE VARCHAR(80) NOT NULL,
  YR NUMERIC(4),
  COUNTRY CHAR(2),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  DIRECTORID INTEGER
)
```

Creating the Example Tables

Example (creating the PERSON table)

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


Creating the Example Tables

Example (creating the CASTING table)

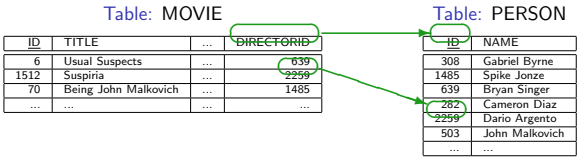
```
CREATE TABLE CASTING (
  MOVIEID INTEGER,
  ACTORID INTEGER,
  ORD INTEGER,
  PRIMARY KEY (MOVIEID, ACTORID)
)
```


Foreign Keys

Definition
foreign key:
the attribute of a relation is the candidate key of another relation

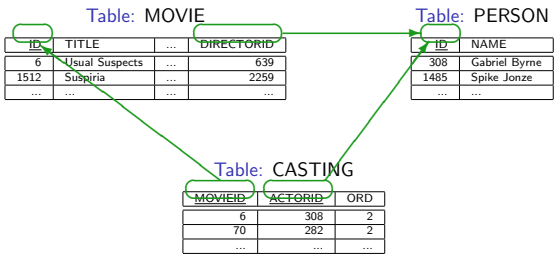
Foreign Key Example

Example (DIRECTORID attribute of MOVIE relation)



Foreign Key Example

Example (foreign keys in 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

- what if a data manipulation request conflicts with referential integrity?

Referential Integrity Example

Example

Table: MOVIE

ID	TITLE	...	DIRECTORID
...	...	...	...
1512	Suspiria	...	2259
...	...	...	...

Table: PERSON

ID	NAME
...	...
2259	Dario Argento
...	...

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

Integrity Constraints

- 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)

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

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 corresponding tuples in the MOVIE relation

Referential Integrity Example

Example (cascade when deleting)

- delete a director from the PERSON relation
- delete all tuples from the MOVIE relation where the ID attribute of the deleted director has the same value as the DIRECTORID attribute
- for each deleted MOVIE tuple, delete all tuples from the CASTING relation where the ID attribute of the deleted movie has the same value as the MOVIEID attribute

Defining Foreign Keys

Statement

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


Foreign Key Definition Example

Example (creating the MOVIE table)

```
CREATE TABLE MOVIE (
   ID INTEGER PRIMARY KEY,
   TITLE VARCHAR(80) NOT NULL,
   ...
   DIRECTORID INTEGER,
   FOREIGN KEY DIRECTORID REFERENCES PERSON (ID)
     ON DELETE RESTRICT
     ON UPDATE CASCADE
)
```

Defining Foreign Keys

- if the foreign key is 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 attribute, it can be specified in the column definition:
  column_name data_type **REFERENCES** table_name [(column_nam

Creating the Example Tables

Example (creating the MOVIE table)

```
CREATE TABLE MOVIE (
  ID INTEGER PRIMARY KEY,
  TITLE VARCHAR(80) NOT NULL,
  YR NUMERIC(4),
  COUNTRY CHAR(2),
  SCORE FLOAT,
  VOTES INTEGER DEFAULT 0,
  DIRECTORID INTEGER REFERENCES PERSON
)
```


Creating the Example Tables

Example (creating the PERSON table)

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


Creating the Example Tables

Example (creating the CASTING table)

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


References

Required text: 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