

Database Systems

Relational Model

H. Turgut Uyar Şule Öğüdücü

2002-2015

1 / 104

License



© 2002-2015 T. Uyar, Ş. Öğüdücü

You are free to:

- Share – copy and redistribute the material in any medium or format
- Adapt – remix, transform, and build upon the material

Under the following terms:

- Attribution – You must give appropriate credit, provide a link to the license, and indicate if changes were made.
- NonCommercial – You may not use the material for commercial purposes.
- ShareAlike – If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

For more information:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Read the full license:

<https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

2 / 104

Topics

Relational Model

Introduction
Keys
Referential Integrity
TutorialD

SQL

Data Types
Data Definition
Data Manipulation
Referential Integrity

3 / 104

Relational Model

- by Dr. E. F. Codd, 1970
- data is modelled as **relations**:
 $\alpha \subseteq A \times B \times C \times \dots$
- 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 can be represented using tables
- relation \rightarrow table, tuple \rightarrow row, attribute \rightarrow column

4 / 104

Relation Example

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

- ▶ relation variable: MOVIE
- ▶ tuple: (Suspiria, 1977, Dario Argento, 7.1, 1004)
- ▶ attribute: YEAR

5 / 104

Relation Structure

- ▶ relation header: set of attributes of the relation
- ▶ affected by data definition language statements
- ▶ relation body: set of tuples in the relation
- ▶ affected by data manipulation language statements

6 / 104

Relation Predicate

- ▶ **relation predicate**: “meaning” of the relation

example

- ▶ “The movie titled TITLE was released in YEAR.
It was directed by DIRECTOR.
The average of VOTES votes is SCORE.”

7 / 104

Tuple Values

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

example: MOVIE relation

- ▶ (Suspiria, 1977, Dario Argento, 1004, 7.1) is True
- ▶ (Suspiria, 1978, Dario Argento, 1004, 7.1) is False

8 / 104

Tuple Order

- ▶ tuple order is insignificant

example

- ▶ these relations are equivalent:

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

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

9 / 104

Attribute Order

- ▶ attribute order is insignificant

example

- ▶ these relations are equivalent:

TITLE	YEAR	...
The Usual Suspects	1995	...
Suspiria	1977	...

YEAR	TITLE	...
1995	The Usual Suspects	...
1977	Suspiria	...

10 / 104

Duplicate Tuples

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

example

TITLE	YEAR	DIRECTOR	SCORE	VOTES
The 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
...



11 / 104

Domains

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

12 / 104

Domain 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 "Suspiria" to DIRECTOR is valid in terms of data types but it doesn't make sense
- ▶ YEAR and VOTES are integers but it doesn't make sense to compare them

13 / 104

Attribute Values

- ▶ attribute values must be scalar
- ▶ no arrays, lists, records, ...

example: multiple directors

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

14 / 104

Null Value

- ▶ value of attribute not known for tuple
- ▶ tuple does not have a value for attribute

example

- ▶ director of movie not known

example

- ▶ nobody voted for movie, therefore no SCORE

15 / 104

Default Value

- ▶ a default value can be used instead of null
- ▶ 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

16 / 104

Keys

- ▶ let B be the set of all attributes of the relation, and let $A \subseteq B$
- ▶ to be a candidate key, A has to be:
- ▶ **unique**: no two tuples have the same values for all attributes in A
- ▶ **irreducible**: no subset of A is unique
- ▶ every relation has at least one candidate key

17 / 104

Candidate Key Examples

- ▶ $\{\text{TITLE}\}$?
- ▶ $\{\text{TITLE}, \text{YEAR}\}$?
- ▶ $\{\text{TITLE}, \text{DIRECTOR}\}$?
- ▶ $\{\text{TITLE}, \text{YEAR}, \text{DIRECTOR}\}$?

18 / 104

Surrogate Keys

- ▶ if a **natural key** can not be found a **surrogate key** can be defined
- ▶ identity attribute
- ▶ its value doesn't matter
- ▶ it can be generated by the system

19 / 104

Surrogate Key Example

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

- ▶ $\{\text{MOVIE#}\}$ is a candidate key
- ▶ $\{\text{MOVIE#}, \text{TITLE}\}$ is not a candidate key

20 / 104

Primary Key

- ▶ if more than one candidate key, one is selected as the **primary key**
- ▶ others are alternate keys
- ▶ names of attributes in the primary key are underlined
- ▶ any attribute that is part of the primary key can not be empty in any tuple
- ▶ every relation must have a primary key

21 / 104

Primary Key Example

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

22 / 104

Scalarity Example

- ▶ how to store actor data?

MOVIE

<u>MOVIE#</u>	TITLE	...	ACTORS
6	The Usual Suspects	...	Gabriel Byrne
...
70	Being John Malkovich	...	Cameron Diaz, John Malkovich
...

23 / 104

Scalarity Example

- ▶ for scalarity, tuples have to be repeated

MOVIE

<u>MOVIE#</u>	TITLE	...	ACTOR
6	The Usual Suspects	...	Gabriel Byrne
...
70	Being John Malkovich	...	Cameron Diaz
70	Being John Malkovich	...	John Malkovich
...

24 / 104

Scalarity Example

MOVIE		
<u>MOVIE#</u>	TITLE	...
6	The Usual Suspects	...
1512	Suspiria	...
70	Being John Malkovich	...
...

ACTOR	
<u>ACTOR#</u>	NAME
308	Gabriel Byrne
282	Cameron Diaz
503	John Malkovich
...	...

CASTING		
<u>MOVIE#</u>	<u>ACTOR#</u>	ORD
6	308	2
70	282	2
70	503	14
...

25 / 104

Scalarity Example

MOVIE			
<u>MOVIE#</u>	TITLE	...	<u>DIRECTOR#</u>
6	The Usual Suspects	...	639
1512	Suspiria	...	2259
70	Being John Malkovich	...	1485
...

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

<u>MOVIE#</u>	<u>ACTOR#</u>	ORD
6	308	2
70	282	2
70	503	14
...

26 / 104

Foreign Keys

- **foreign key**: an attribute of a relation that is a candidate key of another relation

27 / 104

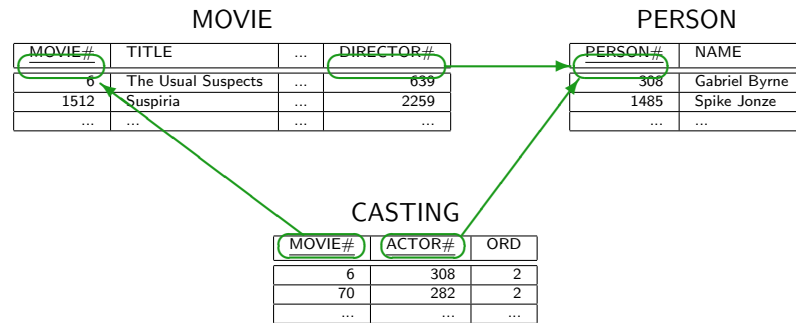
Foreign Key Example

MOVIE			
<u>MOVIE#</u>	TITLE	...	<u>DIRECTOR#</u>
6	The Usual Suspects	...	639
1512	Suspiria	...	2259
70	Being John Malkovich	...	1485
...

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

28 / 104

Foreign Key Examples



29 / 104

Referential Integrity

- ▶ **referential integrity:**
all values of a foreign key attribute must be present among the values of the referenced candidate key attribute
- ▶ if a request would break referential integrity:
- ▶ don't allow
- ▶ reflect the change to affected tuples
- ▶ assign null value
- ▶ assign default value

30 / 104

Referential Integrity Examples

MOVIE				PERSON	
MOVIE#	TITLE	...	DIRECTOR#	PERSON#	NAME
...
1512	Suspiria	...	2259	2259	Dario Argento
...

- ▶ delete (2259, Dario Argento)
- ▶ update (2259, Dario Argento) as (2871, Dario Argento)

31 / 104

Tutorial D Data Types

- ▶ INTEGER
- ▶ RATIONAL
- ▶ BOOL
- ▶ CHAR

32 / 104

Type Definition

- ▶ defining a new type:

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

- ▶ deleting a type:

```
DROP TYPE type_name;
```

33 / 104

Type Definition Examples

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

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

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

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

34 / 104

Type Operations

- ▶ generating a value for a type:

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

example

- ▶ generating a SCORE value:

```
SCORE(8.7)
```

35 / 104

Type Operations

- ▶ getting the value of a field: **THE_** operators

```
THE_field_name(variable_name)
```

example

- ▶ getting the VALUE field of a SCORE variable:

```
THE_VALUE(SCORE)
```

36 / 104

Type Operations

- ▶ type casting: `CAST_AS_` operators
`CAST_AS_target_type(value)`

example

- ▶ casting an integer `VOTES` value to a `RATIONAL`:
`CAST_AS_RATIONAL(VOTES)`

37 / 104

Relation Definition

- ▶ defining a new relation:

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

38 / 104

Relation Definition Example

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

39 / 104

Relation Variables

- ▶ defining a new relation variable

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

- ▶ deleting a relation variable:

```
DROP VAR relvar_name;
```

40 / 104

Relation Variable Examples

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

41 / 104

Relation Variable Examples

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

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

42 / 104

Tuple Generation

- ▶ generating a tuple:

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

43 / 104

Tuple Generation Examples

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

```
TUPLE
{ PERSON# PERSON#(639),
  NAME "Bryan Singer" }
```

44 / 104

Relation Generation

- ▶ generating a relation:

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

- ▶ assigning a relation to a relation variable:

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

45 / 104

Relation Assignment Example

```
MOVIE := RELATION
{ TUPLE
  { MOVIE# MOVIE#(6), TITLE "The Usual Suspects",
    YEAR YEAR(1995), DIRECTOR# PERSON#(639),
    SCORE SCORE(8.7), VOTES 35027 },
  TUPLE
  { MOVIE# MOVIE#(70), TITLE "Being John Malkovich",
    YEAR YEAR(1999), DIRECTOR# PERSON#(1485),
    SCORE SCORE(8.3), VOTES 13809 } };
```

46 / 104

Tuple Insertion

- ▶ inserting tuples:

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

47 / 104

Tuple Insertion Example

```
INSERT MOVIE RELATION
{ TUPLE
  { MOVIE# MOVIE#(1),
    TITLE "Star Wars",
    YEAR YEAR(1977),
    DIRECTOR# PERSON#(360),
    SCORE SCORE(8.8),
    VOTES 53567 } };
```

48 / 104

Tuple Deletion

- ▶ deleting tuples:

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

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

49 / 104

Tuple Deletion Example

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

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

50 / 104

Tuple Update

- ▶ updating tuples:

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

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

51 / 104

Tuple Update 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  
);
```

52 / 104

Attribute Renaming

- ▶ renaming an attribute:

```
RENAME { attribute_name AS new_name }
```

example

- ▶ renaming the DIRECTOR# attribute:

```
RENAME { DIRECTOR# AS PERSON# }
```

53 / 104

Foreign Key Definition

- ▶ defining a foreign key:

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

- ▶ attribute names have to match (rename if necessary)

54 / 104

Foreign Key Examples

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

55 / 104

Foreign Key Examples

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

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

56 / 104

Data Types

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

57 / 104

String Data Types

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

58 / 104

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

59 / 104

Large Object Data Types

- ▶ arbitrary length objects
- ▶ binary: BINARY LARGE OBJECT (n)
- ▶ BLOB
- ▶ text: CHARACTER LARGE OBJECT (n)
- ▶ CLOB
- ▶ can not be used in queries

60 / 104

Domain Creation

- ▶ creating a domain:

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

- ▶ deleting domains:

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

61 / 104

Domain Example

- ▶ a domain for valid SCORE values:

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

62 / 104

Table Creation

- ▶ creating a table:

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

- ▶ deleting tables:

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

63 / 104

Table Creation Example

- ▶ using a domain:

```
CREATE TABLE MOVIE (
  ID INTEGER,
  TITLE VARCHAR(80),
  YR NUMERIC(4),
  DIRECTORID INTEGER,
  SCORE FLOAT,
  VOTES INTEGER
)
```

```
CREATE TABLE MOVIE (
  ID INTEGER,
  TITLE VARCHAR(80),
  YR NUMERIC(4),
  DIRECTORID INTEGER,
  SCORE SCORES,
  VOTES INTEGER
)
```

64 / 104

Null and Default Values

- ▶ defining nullable columns and default values:

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

- ▶ NULL: the column is allowed to be empty (default)
- ▶ NOT NULL: the column is not allowed to be empty

65 / 104

Table Creation Example

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

66 / 104

Value Constraints

- ▶ defining constraints on values:

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

67 / 104

Value Constraint Example

- ▶ SCORE values must be between 1.0 and 10.0

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

68 / 104

Primary Keys

- ▶ defining primary keys:

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

69 / 104

Primary Key Example

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

70 / 104

Primary Keys

- ▶ if the primary key consists of a single column,
it can be specified in column definition:

column_name data_type PRIMARY KEY

example

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

71 / 104

Automatically Incremented Values

- ▶ no standard on defining automatically incremented values
- ▶ PostgreSQL: SERIAL data type
ID SERIAL PRIMARY KEY
- ▶ MySQL: AUTO_INCREMENT property
ID INTEGER PRIMARY KEY AUTO_INCREMENT
- ▶ SQLite: AUTOINCREMENT property
ID INTEGER PRIMARY KEY AUTOINCREMENT

72 / 104

Uniqueness

- ▶ defining unique columns:

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

- ▶ null values are ignored

73 / 104

Uniqueness Example

- ▶ titles and (director, year) pairs are unique:

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

74 / 104

Uniqueness

- ▶ if the uniqueness constraint consists of a single column, it can be specified in the column definition:

column_name data_type **UNIQUE**

example: person names are unique

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

75 / 104

Indexes

- ▶ creating an index

```
CREATE [ UNIQUE ] INDEX index_name  
    ON table_name (column_name [, ...])
```

- ▶ speeds up queries
- ▶ slows down inserts and updates

example: create a year index on movies

```
CREATE INDEX MOVIE_YEAR ON MOVIE (YR)
```

76 / 104

Renaming Tables

- ▶ renaming a table:

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

example

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

77 / 104

Adding Columns

- ▶ adding columns to an existing table:

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

example

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

78 / 104

Deleting Columns

- ▶ deleting columns from a table:

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

example

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

79 / 104

Renaming Columns

- ▶ renaming a column:

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

example

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

80 / 104

Column Defaults

- ▶ setting a default value for a column:

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

- ▶ removing the default value from a column:

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

81 / 104

Adding Constraints

- ▶ adding a new constraint to a table:

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

- ▶ removing a constraint from a table:

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

- ▶ when adding constraints, what happens with existing tuples?

82 / 104

Constraint Addition 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
```

83 / 104

Row Insertion

- ▶ inserting a row to a table:

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

- ▶ order of values must match order of columns
- ▶ if column names are omitted, values must be in order of definition
- ▶ omitted columns will take their default values
- ▶ automatically generated columns are usually omitted

84 / 104

Row Insertion Example

```
INSERT INTO MOVIE VALUES (  
    6,  
    'The Usual Suspects',  
    1995,  
    639,  
    8.7,  
    35027  
)
```

85 / 104

Row Insertion Example

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

- ▶ value for ID will be automatically generated

86 / 104

Row Deletion

- ▶ deleting rows:

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

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

87 / 104

Row Deletion Example

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

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

88 / 104

Row Update

- ▶ updating rows:

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

- ▶ if no condition is specified, all rows will be updated
- ▶ order of column assignments is insignificant

89 / 104

Row Update 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')
```

90 / 104

Foreign Keys

- ▶ defining foreign keys:

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

91 / 104

Foreign Key Example

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

92 / 104

Foreign Keys

- ▶ if the foreign key consists of only one column, it can be specified in the column definition:

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

example

```
CREATE TABLE MOVIE (
    ID SERIAL PRIMARY KEY,
    ...
    DIRECTORID INTEGER REFERENCES PERSON (ID),
    ...
)
```

93 / 104

Foreign Keys

- ▶ if the foreign key refers to the primary key, the referred column can be omitted

example

```
CREATE TABLE MOVIE (
    ID SERIAL PRIMARY KEY,
    ...
    DIRECTORID INTEGER REFERENCES PERSON,
    ...
)
```

94 / 104

Integrity Violation Options

- ▶ what to do if referential integrity will be broken?
- ▶ don't allow if used: RESTRICT, NO_ACTION
- ▶ reflect the change to affected tuples: CASCADE
- ▶ assign null value: SET NULL
- ▶ assign default value: SET DEFAULT

95 / 104

Foreign Keys

- ▶ integrity violation options:

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

96 / 104

Foreign Key Example

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

97 / 104

Referential Integrity Example

MOVIE			
ID	TITLE	...	DIRECTORID
6	The Usual Suspects	...	639
70	Being John Malkovich	...	1485
107	Batman & Robin	...	105

PERSON	
ID	NAME
308	Gabriel Byrne
1485	Spike Jonze

- ▶ MOVIE.DIRECTORID: ON DELETE RESTRICT
- ▶ delete Spike Jonze from PERSON: not allowed
- ▶ delete Gabriel Byrne from PERSON: allowed

98 / 104

Referential Integrity Example

MOVIE		
ID	TITLE	DIRECTORID
6	The Usual Suspects	639
70	Being John Malkovich	1485
107	Batman & Robin	105
112	Three Kings	1070

PERSON	
ID	NAME
308	Gabriel Byrne
1485	Spike Jonze

CASTING		
MOVIEID	ACTORID	ORD
6	308	2
70	282	2
112	1485	4

- ▶ MOVIE.DIRECTORID: ON DELETE CASCADE
- ▶ CASTING.MOVIEID: ON DELETE CASCADE
- ▶ CASTING.ACTORID: ON DELETE CASCADE
- ▶ delete Spike Jonze from PERSON: which rows get deleted?

99 / 104

Referential Integrity Example

MOVIE		
ID	TITLE	DIRECTORID
6	The Usual Suspects	639
70	Being John Malkovich	1485
107	Batman & Robin	105
112	Three Kings	1070

PERSON	
ID	NAME
308	Gabriel Byrne
1485	Spike Jonze

CASTING		
MOVIEID	ACTORID	ORD
6	308	2
70	282	2
112	1485	4

- ▶ MOVIE.DIRECTORID: ON DELETE RESTRICT
- ▶ CASTING.MOVIEID: ON DELETE CASCADE
- ▶ CASTING.ACTORID: ON DELETE CASCADE
- ▶ delete Spike Jonze from PERSON: which rows get deleted?

100 / 104

Example Database

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

101 / 104

Example Database

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

102 / 104

Example Database

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

103 / 104

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

104 / 104