# Database Systems

## Relational Model

H. Turgut Uyar Şule Öğüdücü

2002-2015

## License

## Topics

## Relational Model

- by Dr. E. F. Codd, 1970
- 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 can be represented using tables
- relation $\rightarrow$ table, tuple $\rightarrow$ row, attribute $\rightarrow$ column

## 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

## 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

## 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`."

## 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

## 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 | ... |

## 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 | ... |

## 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 |
| ... | ... | ... | ... | ... |

## 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

## 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

## Attribute Values

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

example: multiple directors

| TITLE | ... | DIRECTORS | ... |
|---|---|---|---|
| ... | ... | ... | ... |
| The Matrix | ... | ~~Andy Wachowski, Lana Wachowski~~ | ... |
| ... | ... | ... | ... |

## Null Value

- value of attribute not known for tuple

example

- director of movie not known

- tuple does not have a value for attribute

example

- nobody voted for movie, therefore no SCORE

## 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`

## 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

## Candidate Key Examples

- `{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 doesn't matter
- it can be generated by the system

## 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 | ... | ... |
| ... | ... | ... | ... | ... | ... |

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

## 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

## Primary 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 | ... | ... |
| ... | ... | ... | ... | ... | ... |

## Scalarity Example

- how to store actor data?

MOVIE

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

## Scalarity Example

- for scalarity, tuples have to be repeated

MOVIE

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

## Scalarity Example

MOVIE

| MOVIE# | TITLE | ... |
|---|---|---|
| 6 | The 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 |
| ... | ... | ... |

## Scalarity Example

MOVIE

| MOVIE# | TITLE | ... | DIRECTOR# |
|---|---|---|---|
| 6 | The 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 |
| ... | ... |

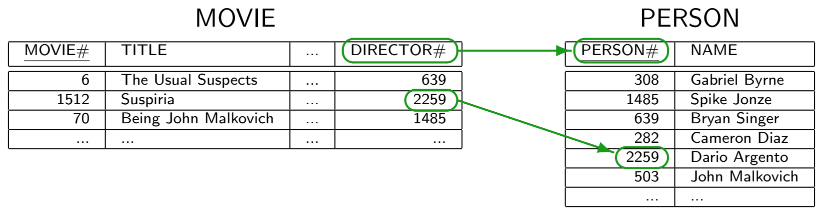
CASTING

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

## Foreign Keys

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

## Foreign Key Example



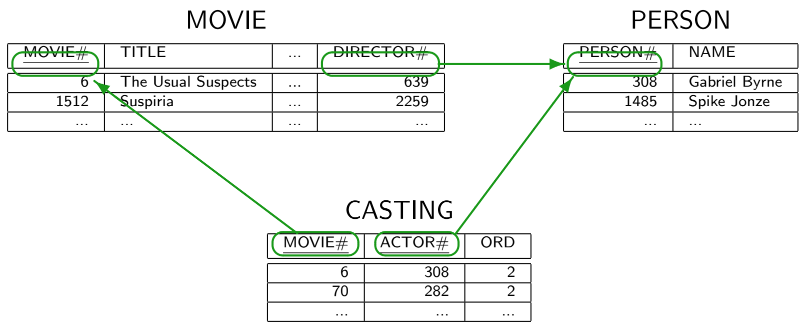
MOVIE

| MOVIE# | TITLE | ... | DIRECTOR# |
|---|---|---|---|
| 6 | The 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 Examples



MOVIE

| MOVIE# | TITLE | … | DIRECTOR# |
|---|---|---|---|
| 6 | The Usual Suspects | … | 639 |
| 1512 | Suspiria | … | 2259 |
| … | … | … | … |

PERSON

| PERSON# | NAME |
|---|---|
| 308 | Gabriel Byrne |
| 1485 | Spike Jonze |
| … | … |

CASTING

| MOVIE# | ACTOR# | ORD |
|---|---|---|
| 6 | 308 | 2 |
| 70 | 282 | 2 |
| … | … | … |

## 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

## Referential Integrity Examples

MOVIE

| MOVIE# | TITLE | … | DIRECTOR# |
|---|---|---|---|
| … | … | … | … |
| 1512 | Suspiria | … | 2259 |
| … | … | … | … |

PERSON

| PERSON# | NAME |
|---|---|
| … | … |
| 2259 | Dario Argento |
| … | … |

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

## Tutorial D Data Types

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

## Type Definition

- defining a new type:

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

- deleting a type:

```
DROP TYPE type_name;
```

## 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) };
```

## Type Operations

- generating a value for a type:

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

example

- generating a SCORE value:

```
SCORE(8.7)
```

## 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)
```

## 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)
  ```

## Relation Definition

- defining a new relation:

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

## Relation Definition Example

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

## Relation Variables

- defining a new relation variable

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

- deleting a relation variable:

  ```
  DROP VAR relvar_name;
  ```

## Relation Variable Examples

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

## 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# };
```

## Tuple Generation

- generating a tuple:

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

## 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" }
```

## Relation Generation

- generating a relation:

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

- assigning a relation to a relation variable:

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

## 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 } };
```

## Tuple Insertion

- inserting tuples:

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

## 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 } };
```

## Tuple Deletion

- deleting tuples:

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

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

## 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));
```

## Tuple Update

- updating tuples:

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

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

## 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
  );
```

## Attribute Renaming

- renaming an attribute:

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

example

- renaming the DIRECTOR# attribute:

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

## 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)

## Foreign Key Examples

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

## Foreign Key Examples

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

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

## 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

## 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)

## 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
- binary: BINARY LARGE OBJECT (n)
- BLOB
- text: CHARACTER LARGE OBJECT (n)
- CLOB
- can not be used in queries

## Domain Creation

- creating a domain:

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

- deleting domains:

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

## Domain Example

- a domain for valid SCORE values:

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

## Table Creation

- creating a table:

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

- deleting tables:

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

## Table Creation Example

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

- using a domain:

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

## 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

## Table Creation Example

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

## Value Constraints

- defining constraints on values:

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

## 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))
)
```

## Primary Keys

- defining primary keys:

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

## 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)
)
```

## 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
)
```

## 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`

## Uniqueness

- defining unique columns:

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

- null values are ignored

## 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)
)
```

## 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
)
```

## 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)
```

## Renaming Tables

- renaming a table:

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

example

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

## 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
```

## Deleting Columns

- deleting columns from a table:

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

example

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

## 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
```

## 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
```

## 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?

## Constraint Addition Example

- YR values can not be less than 1887

```
ALTER TABLE MOVIE
  ADD CONSTRAINT MINIMUM_YEAR
    CHECK (YR >= 1887)
```

- drop the minimum year constraint

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

## 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

## Row Insertion Example

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

## Row Insertion Example

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

- value for `ID` will be automatically generated

## Row Deletion

- deleting rows:

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

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

## 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))
  ```

## 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

## 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')
```

## Foreign Keys

- defining foreign keys:

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

## 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)
)
```

## 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),
  ...
)
```

## 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,
  ...
)
```

## 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

## Foreign Keys

- integrity violation options:

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

## Foreign Key Example

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

## 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

## 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?

## 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?

## 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
)
```

## Example Database

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

## Example Database

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