Relational Datenbases

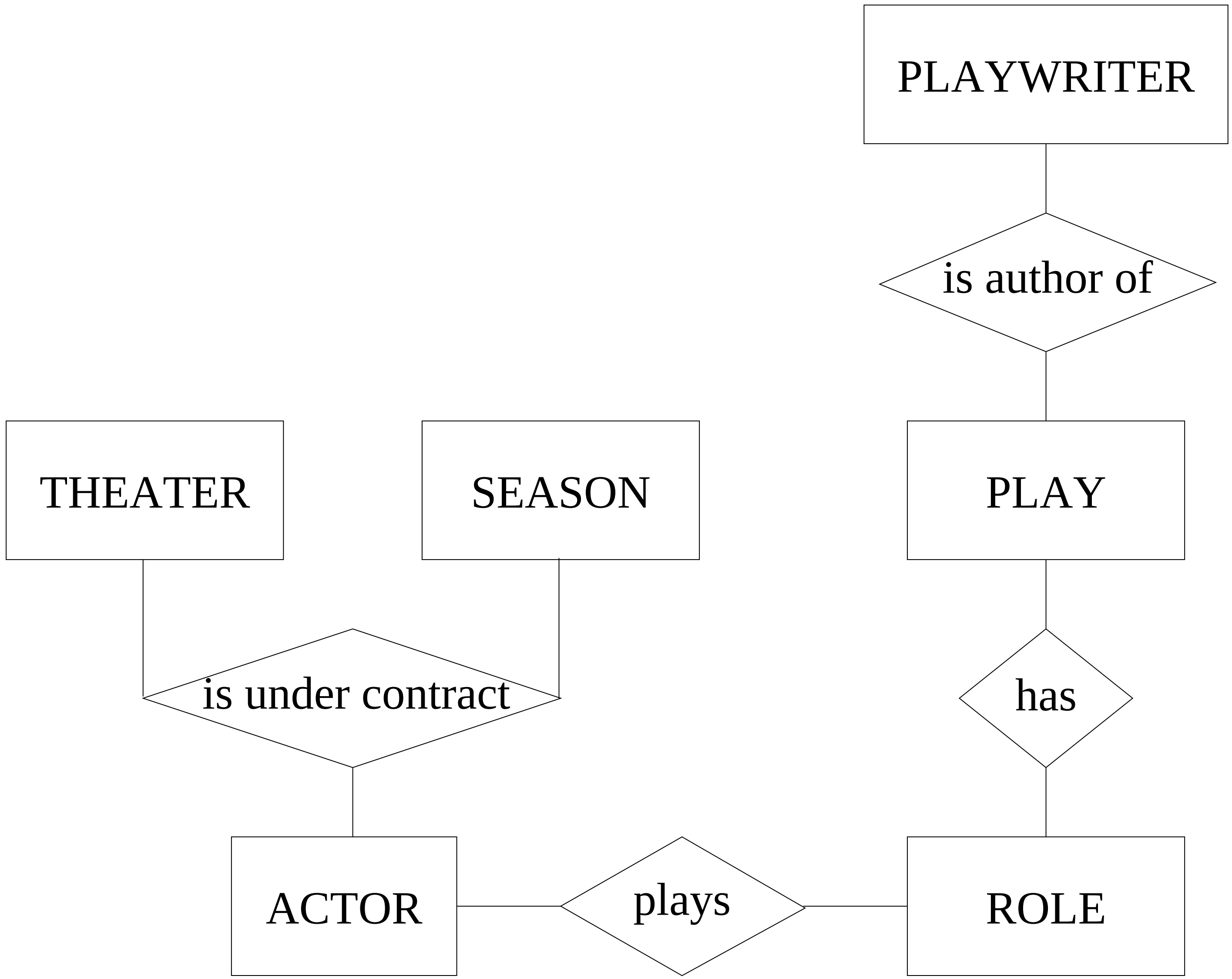
- Data stored as two-dimensional tables,
- A row is a data item, a column is a field.
- A key is a field (or set of fields) that identifies a data item.

Example: Table PLAYWRITERS

<u>AUTHOR</u>	PLACE OF BIRTH	YEAR OF BIRTH
Schiller	Marbach	1759
Goethe	Frankfurt (Main)	1749
Calderón	Madrid	1600
Shakespeare	Stratford	1564
von Kleist	Frankfurt (Oder)	1777

The field AUTHOR is the key

Entity-Relationship Diagramms



Relational scheme

- THEATER: <u>TNAME</u>, CITY
- SEASON: <u>SEASON</u>, <u>YEAR</u>, DURATION
- ACTOR: <u>ID</u>, NAME, CITY
- CONTRACT: <u>ID</u>, <u>TNAME</u>, <u>SEASON</u>, <u>YEAR</u>
- ROLES: <u>CHARACTER</u>, <u>TITLE</u>, TYPE
- PLAYER: <u>ID</u>, <u>CHARACTER</u>, <u>YEAR</u>, <u>TNAME</u>
- PLAY: <u>TITLE</u>, PREMIERE_YEAR, PREMIERE_PLACE
- PLAYWRITER: <u>AUTHOR</u>, BIRTHPLACE, BIRTH_YEAR

Queries

Q1: List all plays (with TITLE, AUTHOR, YEAR) whose premiere took place after 1800.

Q2: Find all actors (NAME, CITY) that have played in some production of “Macbeth”.

Q3: Find all actors (NAME, CITY) that have played in their own city a leading role in some play that premiered in Weimar.

Standard Query Language (SQL)

Basic query:

$$\begin{array}{ll} \textbf{SELECT} & \textsf{AUTHOR} \\ \textbf{FROM} & \textsf{PLAYWRITER} \\ \textbf{WHERE} & \textsf{BIRTHPLACE} = \text{`}\textsf{Madrid}\text{'} \end{array}$$

SQL-Query for Q3

```
SELECT  A.NAME, A.CITY
  FROM  ACTOR A, PLAYER P, ROLE R,
        PLAY PY
 WHERE  A.ID = P.ID
   AND  P.CHARACTER = R.CHARACTER
   AND  R.TITLE = P.TITLE
   AND  PY.PREMIERE_PLACE = 'Weimar'
   AND  R.TYPE = 'Leading'
   AND  PY.PREMIERE_PLACE = A.CITY
```

Connection to predicate logic

- Table $\longrightarrow$ Predicate symbol of arity $=$ number of fields

  PLAYWRITER $\longrightarrow$ $Playwriter(author, birthplace, birth\_year)$

- Data items $\longrightarrow$ Structure $\mathcal{A}$

  $$Playwriter^{\mathcal{A}} = \{\ (\text{Schiller}, \text{Marbach}, 1759), \\ \ldots \\ (\text{vonKleist}, \text{Frankfurt(Oder)}, 1777)\}$$

- SQL-query $\longrightarrow$ Formula with free variables $F(x_1, \ldots, x_n)$

SELECT AUTHOR
FROM PLAYWRITER
WHERE BIRTHPLACE = 'Madrid'

$$Answ(author) = \exists\, birth\_year : Playwriter(author, \text{'Madrid'}, birth\_year)$$

- Answer $\rightarrow$ set of all authors au such that $\mathcal{A}(Answ(au)) = 1$.

SQL-query for query Q3 (simplified)

SELECT A.NAME, A.CITY
FROM ACTOR A, PLAYER P, ROLE R,
WHERE A.ID = PR.ID
AND P.CHARACTER = R.CHARACTER
AND R.TYPE = 'Leading'

$$\begin{aligned} Answ(name, city) = \; & \exists\, id, char, year, tname, title : \\ & Actor(id, name, city) \wedge \\ & Player(id, char, year, tname) \wedge \\ & Role(char, title, \text{'Leading'}) \end{aligned}$$

Nested queries

- Find all actors (NAME) that played ‘Lady Macbeth’ in 2007

```
SELECT  A.NAME
FROM    ACTOR A
WHERE   (‘Lady_Macbeth’, ‘2007’ ) IN
          SELECT  P.CHARACTER, P.YEAR
          FROM    PLAYER P
          WHERE   P.ID = A.ID
```

- Formula for the inner query:

$$\begin{aligned} \mathit{Answ1}(id) = \; & \exists\, tname : \\ & \mathit{Player}(id, \text{'Lady\_Macbeth'}, 2007, tname) \end{aligned}$$

- Formula for the full query:

$$\begin{aligned} \mathit{Answ}(name) = \; & \exists\, id, city : \\ & \mathit{Actor}(id, name, city) \wedge \mathit{Answ1}(id) \end{aligned}$$

Quantified queries

- Find all actors (NAME) that have played at least once

```
SELECT  A.NAME
FROM    ACTOR A
WHERE   EXISTS
          SELECT  *
          FROM    PLAYER P
          WHERE   P.ID = A.ID
```

- Formula for the inner query:

$$\begin{aligned} \mathit{Ans1}(id) = \; & \exists\, character, year, tname : \\ & \mathit{Player}(id, character, year, tname) \end{aligned}$$

- Formula for the query:

$$\begin{aligned} \mathit{Answ}(name) = \; & \exists\, id, city : \\ & \mathit{Actor}(id, name, city) \wedge \mathit{Ans1}(id) \end{aligned}$$

Quantified queries II

- Find all actors (NAME) that have played all leading roles

$$
\begin{aligned}
Ans(name) = \; & \exists\, id, city : \\
& \quad Actor(id, name, city) \wedge \\
& \quad \forall\, char, title : \\
& \qquad Role(char, title, \text{‘Leading’}) \\
& \qquad\qquad \longrightarrow \\
& \qquad \exists\, year, tname : \\
& \qquad Player(id, char, year, tname)
\end{aligned}
$$

SQL query

```
SELECT  A.NAME
FROM    ACTOR A
WHERE   NOT EXISTS
          SELECT  *
          FROM    ROLE R
          WHERE   NOT EXISTS
                    SELECT  *
                    FROM    PLAYER P
                    WHERE   P.ID = A.ID
                    AND     P.CHAR = R.CHAR
```

Definitions and notations

- We write $\mathbf{x}$ für $\{x_1, \ldots, x_n\}$

  $\exists\, \mathbf{x}$ for $\exists\, x_1 \ldots \exists\, x_n$.
- A relation is a formula with free variables, its arity is the number of free variables.
- $R(\mathbf{x})$ denotes a relation with free variables $\mathbf{x}$.
- A condition is a boolean combination of formulas of the form $x = a$.
- $B(\mathbf{x})$ denotes a condition with free variables $\mathbf{x}$.
- If the variables are clear from the context then we write R or B instead of $R(\mathbf{x})$ or $B(\mathbf{x})$.

Relation Algebra

- A formula $R(\mathbf{x})$ of relation algebra has the form:

$$
\begin{array}{lcl}
Tab(\mathbf{x}) & & \\
\sigma_{B(\mathbf{x}')}(R) & = & R(\mathbf{x}) \wedge B(\mathbf{x}') \quad \text{where } \mathbf{x}' \subseteq \mathbf{x} \\
\pi_{\mathbf{x}'}(R) & = & \exists\, \mathbf{x}''\, R(\mathbf{x}) \quad \text{where } \mathbf{x}' \subseteq \mathbf{x},\ \mathbf{x}'' = \mathbf{x} \setminus \mathbf{x}' \\
(R_1 \cup R_2) & = & R_1(\mathbf{x}) \vee R_2(\mathbf{x}) \\
(R_1 - R_2) & = & R_1(\mathbf{x}) \wedge \neg R_2(\mathbf{x}) \\
(R_1 \times R_2) & = & R_1(\mathbf{x}) \wedge R_2(\mathbf{y}) \\
(R_1 \bowtie_{i=j} R_2) & = & \exists z \;\; R_1(x_1, \ldots, x_{i-1}, z, x_{i+1}, x_n) \wedge \\
 & & \qquad R_2(y_1, \ldots, y_{j-1}, z, y_{j+1}, y_m)
\end{array}
$$

SQL $\longrightarrow$ relation algebra

SELECT AUTHOR
FROM PLAYWRITER
WHERE BIRTHPLACE = 'Madrid'

$$Antw(author) = \pi_{author}(\sigma_{birthplace=\text{'Madrid'}}(Playwriter))$$

Evaluation and optimization

- Compute the relations 'bottom-up' .
- Use equivalence rules to speed up evaluation. (Trivial) Examples:

$$
\begin{array}{rcll}
\sigma_{B_1}(\sigma_{B_2}(R)) & \equiv & \sigma_{B_2}(\sigma_{B_1}(R)) & \\
\pi_{\mathbf{x}}(R) & \equiv & \pi_{\mathbf{x}}(\pi_{\mathbf{y}}(R)) & \text{if } \mathbf{x} \subseteq \mathbf{y} \\
\pi_{\mathbf{x}}(\sigma_{B(\mathbf{y})}(R)) & \equiv & \sigma_{B(\mathbf{y})}(\pi_{\mathbf{x}}(R)) & \text{if } \mathbf{x} \supseteq \mathbf{y} \\
\pi_{\mathbf{x}\cup\mathbf{y}}(R \bowtie_{i=j} S) & \equiv & \pi_{\mathbf{x}}(R) \bowtie_{i=j} \pi_{\mathbf{y}}(S) & \text{if } x_i \notin \mathbf{x} \\
 & & & \text{and } y_j \notin \mathbf{y} \\
\sigma_{B(\mathbf{x})}(R \cup S) & \equiv & \sigma_{B(\mathbf{x})}(R) \cup \sigma_{B(\mathbf{x})}(S) & \\
\pi_{\mathbf{x}}(R \cup S) & \equiv & \pi_{\mathbf{x}}(R) \cup \pi_{\mathbf{x}}(S) &
\end{array}
$$