THE MATHEMATICAL THEORY OF RELATIONAL DATABASES

Joachim Borya

Johannes Kepler Universität

November 30th, 2021

History

- What is SQL?
  - Successor of SEQUEL (*Structured English Query Lanugage*)
  - First database system using SQL: IBM System R (1974)
  - For users in the between of IT specialists and other people with technical backgrounds.

Examples

- Simple SQL database: `sqlite`

Time varying relations and algebra

- Mathematical framework for tables in a relational database
- Questions:
  - What is a relation?
  - What does time-varying mean?
  - How can relations be manipulated?
    - "Cartesian product" $\otimes$
    - Projection π
    - Selection σ
    - Join $\bowtie$

Time varying relations and algebra

- A *relation schema* is a finite ordered set $R = (\{A_i\}_{i=1}^n, \leq)$ of *attribute names*.
- Each attribute name corresponds to a *(simple normal) domain* $\mathrm{dom}(A_i) \subseteq U \in \{\Omega^*, \mathbb{N}\}$, $i \in [n]$.
- The set of tuples $\mathrm{Tup}(R)$ on a relation scheme R is

$$\left\{ t : R \to \bigcup_{i=1}^{n} \mathrm{dom}(A_i) : t(A_i) \in \mathrm{dom}(A_i), i \in [n] \right\}.$$

  Because R is ordered we can associate $\mathrm{Tup}(R)$ with $\bigtimes_{i=1}^{n} \mathrm{dom}(A_i)$.
- A *(normalized) relation* r on a relation scheme R is a finite subset of $\mathrm{Tup}(R)$. We also write $r \in \mathrm{Rel}(R)$ and vice versa $R = \mathrm{Att}(r)$.

Normalization of an unnormalized relation

Pet DB		
Owner	**Name**	**Sepecies**
Alice	Banjo	Dog
Alice	Bo	Iguana
Alice	Bailey	Cat
Bob	Aj	Cat
Bob	Angus	Cat
Charlie	Callie	Dog
Dave	Flower	Spider

```
{
    "Alice": {
        "Dog": ["Banjo"],
        "Iguana": ["Bo"],
        "Cat": ["Bailey"]
    },
    "Bob": {
        "Cat": ["Aj", "Angus"]
    },
    "Charlie": {
        "Dog": ["Callie"]
    },
    "Dave": {
        "Spider": ["Flower"]
    }
}
```

Set operations

Definition

Let $R = \{A_1, \ldots, A_n\}$ and $S = \{B_1, \ldots, B_m\}$ be relation schemes and $r \in \mathrm{Rel}(R), s \in \mathrm{Rel}(S)$ relations, then we call r and s *union compatible* if $n = m$ and $\mathrm{dom}(A_i) = \mathrm{dom}(B_i)$ for every $i \in [n]$.

For union-compatible relations we can execute every set operation as usual.

- $r \cup s := \{t \in \mathrm{Tup}(R) : t \in r \vee t \in s\}$
- $r \cap s := \{t \in \mathrm{Tup}(R) : t \in r \wedge t \in s\}$
- $r \backslash s := \{t \in \mathrm{Tup}(R) : t \in r \wedge t \notin s\}$

Cartesian product

Definition

Let $R = \{A_i\}_{i=1}^n$, $S = \{B_i\}_{i=1}^m$ be relation schemes, $r \in \mathrm{Rel}(R)$, $s \in \mathrm{Rel}(S)$ be relations and $t_r \in r$, $t_s \in s$ be tuples. We define

$$t_r \circ t_s := (t_r(A_1), \ldots, t_r(A_n), t_s(B_1), \ldots, t_s(B_m))$$

and

$$r \otimes s := \{t_r \circ t_s : t_r \in r, t_s \in s\}.$$

The relation $r \otimes s \in \mathrm{Rel}(T)$ is the *cartesian product* of the relations r, s on the relation schema

$$T = \{C_i\}_{i=1}^k := \{A_1, \ldots, A_n, B_1, \ldots, B_m\}.$$

Projection

Definition

Let $R = \{A_i\}_{i=1}^n$ be a relation scheme and $r \in \mathrm{Rel}(R)$ a relation. Let $S := \{A_{i_k}\}_{k=1}^m \subseteq R$ be a subset of attribute names. The *projection* of r on S is defined as

$$\pi_S(r) = \{(t(A_{i_1}), \ldots, t(A_{i_m})) : t \in r\}.$$

We immediately see that

$$\pi_S(r) \subseteq \pi_{\{A_{i_1}\}}(r) \otimes \cdots \otimes \pi_{\{A_{i_m}\}}(r).$$

Functional dependency

Definition

Let $R = \{A_i\}_{i=1}^n$ be a relation scheme and $r \in \mathrm{Rel}(R)$ a relation. Let $S_1 = \{A_{i_k}\}_{k=1}^m$ and $S_2 = \{A_{j_k}\}_{k=1}^l$ subsets of R. We call S_2 *functionally dependent* on S_1 w.r.t. r if

$$\{((t(A_{i_k}))_{k=1}^m, (t(A_{j_k}))_{k=1}^l) : t \in r\} \subseteq \pi_{S_1}(r) \times \pi_{S_2}(r)$$

is a function. In this case we write $S_1 \overset{r}{\to} S_2$. We also define

$$\mathrm{Rel}_{S_1 \to S_2}(R) = \{r \in \mathrm{Rel}(R) : S_1 \overset{r}{\to} S_2\}.$$

Keys, prime attributes and second normal form

Definition

Let $R = \{A_i\}_{i=1}^n$ be a relation schema, $r \in \mathrm{Rel}(R)$ a relation and $K \subset R$. We call K a key of the relation r, if $K \to R \backslash K$ and there is no $K' \subset K$ with $K' \to R \backslash K'$. A superset of a key K is called a superkey and the attributes $A \in K$ of a key are called prime. An attribute $A \in R$ s.t. there is no key K with $A \in K$ is called non-prime.

Definition

Let $R = \{A_i\}_{i=1}^n$ be a relation schema and $r \in \mathrm{Rel}(R)$ a relation. Then we say, that r is in second normal form, if for every non-prime attribute $A \in R$, every key $K \subseteq R$ of r and $S \subset K$, $K \to \{A\}$ and $S \not\to \{A\}$ hold.

Selection

Definition

Let $R = \{A_i\}_{i=1}^n$ a relation schema, $r \in \mathrm{Rel}(R)$ a relation and $a \in \mathrm{dom}(A_\nu)$ for some $A_\nu \in R$. Then we define the *selection* as

$$\sigma_{A_\nu = a}(r) := \{t \in r : t(A_\nu) = a\}.$$

- $\sigma_{A=a}(\sigma_{B=b}(r)) = \sigma_{B=b}(\sigma_{A=a}(r))$
- $\sigma_{A=a}(r \gamma s) = \sigma_{A=a}(r) \gamma \sigma_{A=a}(s)$ for $\gamma \in \{\cup, \cap, \setminus\}$
- $\sigma_{A=a \wedge B=b}(r) = \sigma_{A=a}(r) \cap \sigma_{B=b}(r)$

Join

Definition

Let $R = \{A_i\}_{i=1}^n$, $S = \{B_i\}_{i=1}^m$ be relation schemes, $r \in \mathrm{Rel}(R)$, $s \in \mathrm{Rel}(S)$ be relations and $A_\nu \in R, B_\mu \in S$ attribute names. The relation

$$r \overset{A_\nu = B_\mu}{\bowtie} s := \{t_r \circ t_s : t_r \in r \wedge t_s \in s \wedge t_r(A_\nu) = t_s(B_\mu)\}$$

is called *(equi-)join* of r and s on A_ν and B_μ.

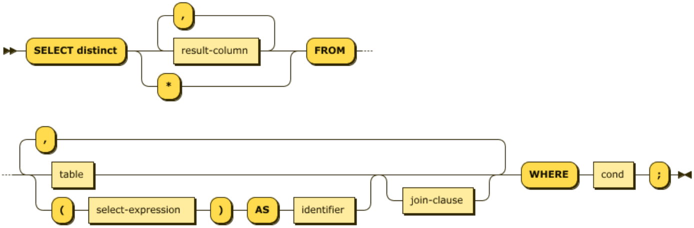
- Selections and joins are exchangeable.

$$r \overset{A_\nu = B_\mu}{\bowtie} s = \sigma_{A_\nu = B_\mu}(r \otimes s)$$

$$r \overset{A_\nu = A}{\bowtie} \{(a)\} = \sigma_{A_\nu = a}(r)$$

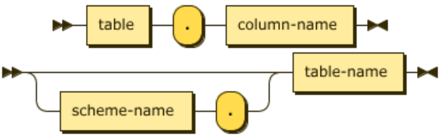
for $a \in \mathrm{dom}(A_\nu)$ and $\{(a)\} \in \mathrm{Rel}(\{A\})$

select-expression



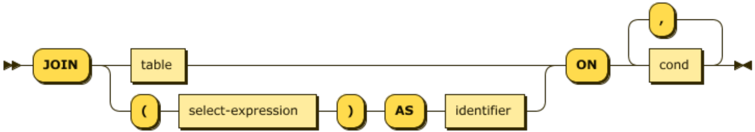
- All valid expressions have this form.
- The query is restricted to produce distinct tuples.
- The part `identifier` is an ad-hoc replacement for a `table-name`.
- The set of all such expression will be called `SPJ`.

result-column, table



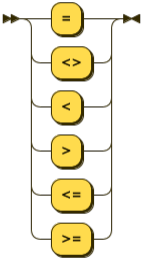
- The table of a column has to be referenced as well (technically SQL automatically looks this up).
- The part `scheme-name` has no importance in our considerations.

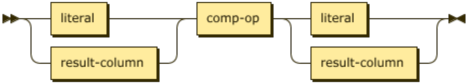
join-clause

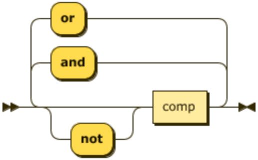


- This clause provides the rhs of a join.

comp-op, comp, cond







- The part `literal` denotes a C-style integer or string literal.
- The part `cond` is a simple boolean expression involving literals, column names and comparison operators.

Meaning of a SPJ expression

- What are the objects we work with? Answer:

$$\mathcal{R} := \left\{ r \subset \bigtimes_{i=1}^{n} D_i : n \in \mathbb{N},\ \forall i \in [n] : D_i \subset \mathbb{N} \vee D_i \subset \Omega^* \right\}$$

- In reality only finite relations are important.

Mapping of tables and relations

- How do we connect the concepts of tables and relations?
- Let $\{R_i\}_{i=1}^m$ be relation schemes, i.e.

$$R_i := \{A_{i1}, \ldots, A_{in_i}\} \text{ for } i \in [m]$$

  and $r_1 \in \mathrm{Rel}(R_1), \ldots, r_m \in \mathrm{Rel}(R_m)$ relations.
- Consider a set of variables e.g. identifiers

$$V := \{\texttt{tid1}, \ldots, \texttt{tidm}, \texttt{tid1.att1}, \ldots, \texttt{tid1.attn1},$$
$$\ldots, \texttt{tidm.att1}, \ldots, \texttt{tidm.attnm}\}$$

  and a map $\mathcal{D} : V \to \{r_i\}_{i=1}^m \cup \bigcup \{R_i\}_{i=1}^m$ given by

$$\mathcal{D}(\texttt{tid1}) := r_1, \ldots, \mathcal{D}(\texttt{tidm}) := r_m,$$
$$\mathcal{D}(\texttt{tid1.att1}) := A_{11}, \ldots, \mathcal{D}(\texttt{tid1.attn1}) := A_{1n_1},$$
$$\ldots, \mathcal{D}(\texttt{tidm.att1}) := A_{m1}, \ldots, \mathcal{D}(\texttt{tidm.attnm}) := A_{mn_m}$$

.

Semantic function definition

- Goal: Construction of a map $[\![\cdot]\!]_{\mathcal{D}} : \mathtt{SPJ} \to \mathcal{R}$.

Semantic function

for `table` expressions t

$[\![\texttt{SELECT * FROM } t]\!]_{\mathcal{D}} := \mathcal{D}(t)$.

for `select-expressions` S

$$[\![\texttt{SELECT * FROM (}S\texttt{) AS subquery}]\!]_{\mathcal{D}} =$$
$$[\![\texttt{SELECT * FROM subquery}]\!]_{\mathcal{D}^*} \text{ with}$$
$$\mathcal{D}^*(E) := \begin{cases} [\![E]\!]_{\mathcal{D}}, & \text{if } E = \texttt{subquery} \\ A, & \text{if } E = \texttt{subquery.*} \\ \mathcal{D}(E), & \text{else} \end{cases}$$

The attribute names of $[\![\texttt{subquery}]\!]_{\mathcal{D}}$ are inherited from the tables involved in the construction of the subquery but with the prefix `subquery.` Note that $A \in \mathit{Att}([\![\texttt{subquery}]\!]_{\mathcal{D}})$.

Example

$Q_S :=$ `SELECT adult.name, adult.address`
`FROM (`S`)`
`) AS adult;`

$S =$ `SELECT owner.name, owner.address`
`FROM owner`
`WHERE owner.age > 17;`

Meaning

$$\mathrm{Att}([\![S]\!]_{\mathcal{D}}) = \{N, A\}$$

$\mathcal{D}^* := \mathcal{D} \cup [\texttt{adult} \mapsto [\![S]\!]_{\mathcal{D}}, \texttt{adult.name} \mapsto N, \texttt{adult.address} \mapsto A]$

$[\![Q_S]\!]_{\mathcal{D}} = [\![\texttt{SELECT adult.name, adult.address FROM adult}]\!]_{\mathcal{D}^*}$

Semantic function

for `table` or `select-expressions` $F_1, \ldots, F_n$

$[\![\texttt{SELECT * FROM } F_1, \ldots, F_n]\!]_{\mathcal{D}} = \bigotimes_{i=1}^{n} [\![\texttt{SELECT * FROM } F_i]\!]_{\mathcal{D}}$

for `result-column` expressions $c_1, \ldots, c_n$

$[\![\texttt{SELECT } c_1, \ldots c_n \texttt{ FROM } F]\!]_{\mathcal{D}} = \pi_{(\mathcal{D}(c_1), \ldots, \mathcal{D}(c_n))}([\![\texttt{SELECT * FROM } F]\!]_{\mathcal{D}})$

It is only defined this way if $\mathcal{D}(c_1), \ldots, \mathcal{D}(c_n) \in Att([\![\texttt{SELECT * FROM } F]\!]_{\mathcal{D}})$.

Semantic function

for a list of `table` or `select-expressions` F

$[\![\texttt{SELECT * FROM } F \texttt{ WHERE } c \texttt{ = } a]\!]_{\mathcal{D}} = \sigma_{\mathcal{D}(c)=\bar{a}}([\![\texttt{SELECT * FROM } F]\!]_{\mathcal{D}})$

It is only defined this way if $\mathcal{D}(c) \in Att(F)$ and $a \in dom(\mathcal{D}(c))$.

- $[\![\texttt{SELECT * FROM } F \texttt{ WHERE } c \texttt{ < } a]\!]_{\mathcal{D}} = \bigcup_{i=1}^{\infty} \sigma_{\mathcal{D}(c)=\bar{a}-i}([\![\texttt{SELECT * FROM } F]\!]_{\mathcal{D}})$
- $[\![\texttt{SELECT * FROM } F \texttt{ WHERE } c \texttt{ <= } a]\!]_{\mathcal{D}} = \bigcup_{i=0}^{\infty} \sigma_{\mathcal{D}(c)=\bar{a}-i}([\![\texttt{SELECT * FROM } F]\!]_{\mathcal{D}})$

Semantic function

for `table` or `select-expressions` F_1, F_2 and $c_1 \in Att(F_1), c_2 \in Att(F_2)$

$$\llbracket \texttt{SELECT * FROM } F_1 \texttt{ JOIN } F_2 \texttt{ ON } c_1 \texttt{ = } c_2 \rrbracket_{\mathcal{D}}$$
$$= \llbracket \texttt{SELECT * FROM } F_1 \rrbracket_{\mathcal{D}} \overset{\mathcal{D}(c_1)=\mathcal{D}(c_2)}{\bowtie} \llbracket \texttt{SELECT * FROM } F_2 \rrbracket_{\mathcal{D}}$$
$$= \sigma_{\mathcal{D}(c_1)=\mathcal{D}(c_2)}(\llbracket \texttt{SELECT * FROM } F_1 \rrbracket_{\mathcal{D}} \otimes \llbracket \texttt{SELECT * FROM } F_2 \rrbracket_{\mathcal{D}})$$
$$= \llbracket \texttt{SELECT * FROM } F_1, F_2 \texttt{ WHERE } c_1 \texttt{ = } c_2 \rrbracket_{\mathcal{D}}$$

Example

```sql
SELECT inventory.desc, inventory.stock
FROM inventory
JOIN (SELECT * FROM product WHERE product.type = "fruit")
AS fruits
ON inventory.desc = fruits.desc
WHERE inventory.stock < 8;
```

Example

$[\![$ SELECT inventory.desc, inventory.stock
FROM inventory
JOIN (SELECT * FROM product WHERE product.type = "fruit") AS fruits
ON inventory.desc = fruits.desc
WHERE inventory.stock < 8 $]\!]_{\mathcal{D}}$

$= \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}($
$[\![$ SELECT * FROM inventory
JOIN (SELECT * FROM product WHERE product.type = "fruit") AS fruits
ON inventory.desc = fruits.desc
WHERE inventory.stock < 8 $]\!]_{\mathcal{D}}$
)

Example

$$= \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}(\sigma_{\mathcal{D}(\texttt{inventory.stock})<8}([\![\texttt{SELECT * FROM inventory JOIN (SELECT * FROM product WHERE product.type = "fruit") AS fruits ON inventory.desc = fruits.desc}]\!]_{\mathcal{D}}))$$

$$= \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}(\sigma_{\mathcal{D}(\texttt{inventory.stock})<8}([\![\texttt{SELECT * FROM inventory}]\!]_{\mathcal{D}} \bowtie_{\mathcal{D}(\texttt{inventory.desc})=\mathcal{D}(\texttt{fruits.desc})} [\![\texttt{SELECT * FROM (SELECT * FROM product WHERE product.type = "fruit") AS fruits}]\!]_{\mathcal{D}}))$$

Example

$$= \ \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}($$
$$\sigma_{\mathcal{D}(\texttt{inventory.stock})<8}(\llbracket \texttt{ SELECT * FROM inventory } \rrbracket_{\mathcal{D}}$$
$$\bowtie_{\mathcal{D}(\texttt{inventory.desc})=\mathcal{D}(\texttt{fruits.desc})}$$
$$\llbracket \texttt{ SELECT * FROM product WHERE product.type = "fruit") } \rrbracket_{\mathcal{D}}$$
$$))$$

$$= \ \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}($$
$$\sigma_{\mathcal{D}(\texttt{inventory.stock})<8}(\ \mathcal{D}(\texttt{inventory})$$
$$\bowtie_{\mathcal{D}(\texttt{inventory.desc})=\mathcal{D}(\texttt{fruits.desc})}$$
$$\sigma_{\mathcal{D}(\texttt{product.type}=\text{"}\textit{fruit}\text{"})}(\mathcal{D}(\texttt{product}))$$
$$))$$

$$= \ \pi_{(\mathcal{D}(\texttt{inventory.desc}),\mathcal{D}(\texttt{inventory.stock}))}($$
$$\bigcup_{i=1}^{\infty} \sigma_{\mathcal{D}(\texttt{inventory.stock})=8-i}(\ \mathcal{D}(\texttt{inventory})$$
$$\bowtie_{\mathcal{D}(\texttt{inventory.desc})=\mathcal{D}(\texttt{fruits.desc})}$$
$$\sigma_{\mathcal{D}(\texttt{product.type}=\text{"}\textit{fruit}\text{"})}(\mathcal{D}(\texttt{product}))$$
$$))$$

Sources

- E. F. Codd, "Further normalization of the database relational model", 1971.
- Gottfried Vossen, "Datenbankmodelle, Datenbanksprachen und Datenbankmangementsysteme", 2008.
- David Maier, "The theory of relational databases", 1983.
- https://sqlite.org/lang_select.html

Syntax diagrams created with Railroad Diagram Generator
https://www.bottlecaps.de/rr/ui