Chapter 5

Theoretical Preliminaries
Chapter 5: Relational Databases

5.1 Predicate Calculus

How does Predicate Calculus allow us to specialize queries non-sequentially?

Why is the relationship between a valid interpretation and a database?

What is the relationship between the level of abstraction and a database?
Chapter 3: Theoretical Foundations

Formulas

1. Each atom is a formula.

2. If \( p \) and \( q \) are formulas, so are \( p \land q \) and \( p \lor q \).

3. \( p \rightarrow q \) is a formula.

4. \( \neg p \) is a formula.

5. Predicates Any symbol so designated may be a predicate, \( (\forall x \exists ! y) \) and \( \forall z \exists y \). (2) Often present \( \neg p \) and \( \exists z \) and \( \forall x \). Predicates Any symbol so designated may be a predicate, \( (\exists ! x \forall y) \) and \( \exists y \forall x \). (2) Often present \( \neg p \) and \( \exists z \) and \( \forall x \).

6. Logical Connectors \( \neg, \land, \lor, \rightarrow, \leftrightarrow \).

7. Quantifiers \( \exists \) and \( \forall \). These symbols are called existential and universal quantifiers respectively, and are read as there exists.

8. Parentheses and \( ( \neg, \land, \lor, \rightarrow, \leftrightarrow \) are punctuation. Some readers find \( \neg p \) with \( \neg \) explicitly written more legible.

Terms

9. Atomic Terms \( \exists (x) \)... \( \exists (y) \) are terms, \( (x) \) and \( (y) \) are terms, \( f (x) \) and \( f (y) \) are terms, \( f (x, y) \) are terms.

Axioms

1. If \( p \) and \( q \) are sentences, we may use the longer form and the shorter form.

2. A term, called a function symbol, the common configuration pred-

3. If \( p \) is an atom, \( \exists (x) \) is a term, \( f (x) \) is a term, \( f (x, y) \) is a term.

4. Each variable and each constant is a term.

5. Product of Functions Any symbol so designated may be a predicate, \( (\exists ! x \forall y) \) and \( \exists y \forall x \)

\( \neg p \) and \( \exists y \) and \( \forall x \) respectively.

\( \neg p \) and \( \exists z \) and \( \forall x \).

\( \neg p \) and \( \exists z \) and \( \forall x \).
The Pedagogical Meanings of the Logical Operations appear in the following table:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implication</td>
<td>T</td>
</tr>
<tr>
<td>Conjunction</td>
<td>T</td>
</tr>
<tr>
<td>Disjunction</td>
<td>F</td>
</tr>
<tr>
<td>Negation</td>
<td>T</td>
</tr>
<tr>
<td>Identity</td>
<td>F</td>
</tr>
<tr>
<td>Implication</td>
<td>T</td>
</tr>
<tr>
<td>Conjunction</td>
<td>T</td>
</tr>
<tr>
<td>Disjunction</td>
<td>T</td>
</tr>
<tr>
<td>Negation</td>
<td>T</td>
</tr>
<tr>
<td>Identity</td>
<td>T</td>
</tr>
</tbody>
</table>

In addition, Table 5 provides some useful equivalences, which we may use to simplify expressions or to convert one form to another more convenient form.

To define the semantics of a WFF, we explain how to evaluate W, so that

5.1.2 Semantics

completely avoid confusion by meaning variables.

For example, consider the WFF

\[(0 = x + y) \land (y < x)\]

where \(x\) is a variable, \(0\) is the subformula \(x\), and the scope of \(x\) is the subformula \(x\).

Before proceeding to order the variables and the scope of variables, we must introduce the WFF

\[
(((z \cdot y) \cdot x) \cdot y) \land ((z \cdot y) \cdot (x \cdot y))
\]

where \(z\) and \(y\) are free in the formule.
### Useful Equivalences and Implications for Formulas with Quantifiers

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x (P(x) \land Q(x)) \Leftrightarrow (\exists x P(x) \land Q(x))$</td>
<td>Universal quantifier distributes over conjunction.</td>
</tr>
<tr>
<td>$\exists x (P(x) \leftrightarrow Q(x)) \Leftrightarrow (\exists x P(x) \lor \exists x Q(x))$</td>
<td>Existential quantifier distributes over equivalence.</td>
</tr>
</tbody>
</table>

### Example

As an example, let the domain $D$ for an interpretation $I$ be the set of all integers, i.e., $D = \{0, 1, 2, \ldots\}$. Then, $\forall x (P(x) \land Q(x)) \Leftrightarrow (\exists x P(x) \land Q(x))$. If $I$ is the standard interpretation, then $\forall x P(x)$ means that every integer is a member of the set $P$.

### N-ary Predicates

Each n-ary predicate is assigned to a predicate from $D^n$ to $T_F$.

### N-ary Functions

Each n-ary function is assigned to a function from $D^n$ to $D$.

### Constants

Each constant is assigned a value in $D$. For constants, we make the unique-name assumption, which means that literal numbers and strings, if used, must be different. User-defined application constants may be assigned to any value in $D$ (e.g., a map may define a constant to be the value 0).
Chapter 2: Theoretical Foundations

188

Office Dynamics Development

5.1.3 Counting Quantifiers

Therefore, we are able to evaluate an open formula over a domain using a counting quantifier, which is now d, e, and can be extended to a structure, which we may now interpret in the context of a domain. For example, consider the open formula: 

Therefore, the results we obtain without the counting quantifiers are strictly a consequence.
5.2.1 Model Theory

Theoretical Foundations of Object-Relational Databases

In this section, we will focus on the object-relational view of a database. We will start by discussing the concept of a multidimensional table and how it relates to the object-relational view. We will then move on to the discussion of the object-relational model and how it is different from the traditional relational model.

5.2 Theoretical Foundations

For example, we may have a set of data that is represented by a table in a relational database. This table may contain information about customers, orders, and products. Each row in the table represents a specific customer order, and each column represents a specific attribute of that order. For example, we may have columns for customer name, order date, and order amount.

In contrast, the object-relational model allows us to represent data in a more flexible way. Instead of using tables, we can use objects to represent data. Each object can have attributes, and these attributes can be related to other objects. For example, we may have an object for a customer, and this object can have attributes for the customer's name, address, and phone number. We can also create relationships between objects, such as a relationship between a customer object and an order object.

The object-relational model also allows us to represent complex data types, such as multimedia data, in a more natural way. For example, we can create objects to represent images, audio files, and video files. We can then use these objects to store and manipulate the multimedia data.

In summary, the object-relational model provides a more flexible and powerful way to represent data than the traditional relational model. It allows us to represent complex data types and relationships in a more natural way.
We say that an interpretation is valid if all the closed formulas are true.

Forums, thus, given an interpretation, they can evaluate to true or false.

We say that a formula is satisfied by a model if the interpretation is a model.

Models are said to be equivalent if they satisfy the same set of formulas.

Models in Model Theory
show an relational calculus.

We assume these more specific representations for the rest of our discussion.

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]

Hence, we can now write a query 1 more succinctly as follows:

\[
\text{there exist } x, y \text{ such that } x < 10 \land y < 10
\]

We infer attribute names for the result table by taking the operator sets from the

\[
\text{less than } 10 \land q(y) \land p(x) \Rightarrow \text{less than } 10 \land q(y) \land p(x)
\]

lines because although \( > \) is not a member of \( \Delta \), \( = \) is a member of \( \Delta \). Hence, \( = \) and \( = \) are in the
depth table. We can now see that we can refocus our attention on the

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]

The relationship of columns for \( x \) and \( y \) does not hold.

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]

When we substitute the game for \( x \) and \( y \) for \( y \) there indeed exists a

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]

match. Let all items that cost more than \( \$15 \) along with their department

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]

query 1. For the more-theoretic approach to databases, open formulas are

\[
\text{less than } 10 \land p(x) \land q(y) \Rightarrow \text{less than } 10 \land p(x) \land q(y)
\]


5.2.2 Relational Calculus

Chapter 5: Theoretical Preliminaries
These ways are slightly incorrect and are worth discussing. Chapter 5 considers

two examples of how to express quirky facts in SQL.

In addition to these two ways of expressing quirky facts in SQL, there
are many ways to express quirky facts in SQL. For example, we might
want to choose the department that costs $10 and any other department
that is not a quirky department. Thus, we can express this

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

We can also express this in SQL by substituting any department

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

and again in the opposite order:

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

Now let's see how to express a quirky fact in SQL.

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

We can also express this in SQL by substituting any department

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

and again in the opposite order:

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]

We can also express this in SQL by substituting any department

\[ \text{SELECT department FROM employee WHERE department \neq \text{quirky department} \land \text{cost} = 10} \]
For Query 6, we want to compare the total income of two different departments and then select the one with the greater income.

\[ \text{<department_income>} \]

\[ \text{query 6: let the managers who are responsible for every item that costs} \]

\[ \text{more than$10} \]

\[ \text{and their income} \]

\[ \text{greater than$0} \]

\[ \text{have the following result:} \]

\[ \text{who yields the following (strange) result:} \]

\[ \text{which yields the following result:} \]

\[ \text{Next consider the query} \]

\[ \text{versal quantifier on a relational expression of an existential quantifier} \]

\[ \text{to fetch all the items in each department which means that we need a null-}
\]

\[ \text{for some item whose name is not cost$10. Thus we must come} \]

\[ \text{to fulfillment. We want departments for which none of the items cost$10 or} \]

\[ \text{both the result because &nrarr; query$10, query$10 is a tuple in d whose price} \]

\[ \text{in the result is greater than$0. Managed. Managed.} \]

\[ \text{which yields the following result:} \]

\[ \text{Query 6. Let the managers who are responsible for every item that costs} \]

\[ \text{more than$10} \]

\[ \text{and their income} \]

\[ \text{greater than$0} \]

\[ \text{have the following result:} \]

\[ \text{which yields the following result:} \]

\[ \text{Next consider the query} \]

\[ \text{versal quantifier on a relational expression of an existential quantifier} \]

\[ \text{to fetch all the items in each department which means that we need a null-}

\[ \text{for some item whose name is not cost$10. Thus we must come} \]

\[ \text{to fulfillment. We want departments for which none of the items cost$10 or} \]

\[ \text{both the result because &nrarr; query$10, query$10 is a tuple in d whose price} \]

\[ \text{in the result is greater than$0. Managed. Managed.} \]
appear in $\mu$ and also appear in at least one of the $p_i$’s is a constant of one of the predicates $\forall x \in \mathcal{X} \ldots x^\mu$ and any variable that appears in a different number of places for $\exists x$ and $\forall x$ reads over at most $n$ symbols of $\mathcal{X}$ and the $p_i$’s and are atomic predicates each of which may

$$\bigwedge_{(0 \vee d \ldots \vee d_i \ldots \vee d)_{\mathcal{X}^{\forall}}, \mathcal{X}^{\exists}} \mathcal{X}$$

Although various deductive database systems have been defined for

another standard view of a database is a proof-theoretic view, which pro-

defines a predicate, we begin, however, by describing the first-order

negeration, which is characterized by closed formulas of the form

For example, the form $\forall x \in \mathcal{X} p_i \leftarrow$ $p_j$ $\in \mathcal{X}$.

defines a predicate, we begin, however, by describing the first-order

Because if not, the resolution of the $\exists x$ and $\forall x$ reads over at most $n$ symbols of $\mathcal{X}$ and the $p_i$’s and are atomic predicates each of which may

$\bigwedge_{(0 \vee d \ldots \vee d_i \ldots \vee d)_{\mathcal{X}^{\forall}}, \mathcal{X}^{\exists}} \mathcal{X}$

Axioms

Another standard view of a database is a proof-theoretic view, which pro-

The other solutions do not have direct translations to $\text{SQL}$, and because of

$\text{SELECT}$

$\text{FROM}$

$\text{WHERE}$

$p \neq 10$ and $m \neq 1$, \text{Deep}