**CS 31 Database Management Programming Lecture 3**

**Functional Dependency** – a relationship between columns in which one column or group of columns determines the value of another

**Determinant** – one or more columns that functionally determine another column or columns

**Functional dependency/Determinant notation**

X → Y is read as X determines Y, or Y is functionally dependent on X

* y = x + 3, x → y
* cookieCost = numberOfBoxes × $5, numberOfBoxes → cookieCost
* extendedPrice = quantity × unitPrice, (quantity, unitPrice) → extendedPrice

**Composite Determinant** – a determinant that consists of more than one column

**Decomposition Rule** – if A → (B, C), then A → B and A → C

**Union Rule** – if A → B and A → C, then A → (B, C)

A **relation** is a two-dimensional array with the following characteristics:

* Rows contain data about an entity
* Columns contain data about attributes of the entities
* All entries in a column are of the same kind
* Each column has a unique name
* Cells of the table hold a single value
* The order of the columns is unimportant
* The order of the rows is unimportant
* No two rows may be identical

**Entity** – something identifiable that users want to track

**Attributes (columns)** – the columns of a relation

**Tuples (rows)** – the rows of a relation

File, field, and record are also used in place of table, column, and row, respectively.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employee Number** | **FirstName** | **LastName** | **Department** | **Email** | **Phone** |
| 100 | Jerry | Johnson | Accounting | jj@someplace.com | 834-1101 |
| 200 | Mary | Abernathy | Finance | ma@someplace.com | 834-2101 |
| 300 | Liz | Smathers | Finance | ls@someplace.com | 834-2102 |
| 400 | Tom | Caruthers | Accounting | tc@someplace.com | 834-1102, 834-1191, 834-1192 |
| 500 | Tom | Jackson | Production | tj@someplace.com | 834-4101 |
| 600 | Eleanore | Caldera | Legal | ec@someplace.com | 834-3101 |
| 700 | Richard | Bandalone | Legal | rb@someplace.com | 834-3102, 834-3191 |

Fig.1 – Nonrelational table (multiple entries per cell)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employee Number** | **FirstName** | **LastName** | **Department** | **Email** | **Phone** |
| 100 | Jerry | Johnson | Accounting | jj@someplace.com | 834-1101 |
| 200 | Mary | Abernathy | Finance | ma@someplace.com | 834-2101 |
| 300 | Liz | Smathers | Finance | ls@someplace.com | 834-2102 |
| 400 | Tom | Caruthers | Accounting | tc@someplace.com | 834-1102 |
|  |  |  |  | Fax: | 834-9911 |
|  |  |  |  | Home: | 723-8795 |
| 500 | Tom | Jackson | Production | tj@someplace.com | 834-4101 |
| 600 | Eleanore | Caldera | Legal | ec@someplace.com | 834-3101 |
|  |  |  |  | Fax: | 834-9912 |
|  |  |  |  | Home: | 723-7654 |
| 700 | Richard | Bandalone | Legal | rb@someplace.com | 834-3102 |

Fig.2 – Nonrelational table (not same data type and row order matters)

**Composite Key** – keys that have two or more columns

**Superkey** – a column or combination of columns that uniquely identifies any row in a relation

**Candidate Key** – a minimal superkey

**Primary Key** – the candidate key defined to the DBMS to be the primary key

The DBMS will use the primary key as its primary means of selecting rows in a relation.

**Surrogate Key** – an artificial column added by the DBMS to serve as the primary key for a table

The DBMS will make sure the value is unique and the assigned value never changes. Surrogate keys are used when a primary key is large. Some database designers believe all primary keys should be surrogate keys.

**Foreign Key** – is a column or composite of columns that is the primary key of another table

It is a key of a table *foreign* to the one in which it appears.

**Domain Integrity Constraint** – a data constraint that limits data to a specific set of values

**Entity Integrity Constraint** - a primary key, whether it’s a single column or composite key, must have unique values for every row in the table

**Referential Integrity Constraint** – a statement that limits the values of the foreign key to matching values of the primary key in the referenced table

**NOTE: NULL** values are also allowed, but allowing **NULL** values for foreign key column(s) should probably be avoided. **NULL** means no value – may exist but not entered, may not exist, may not be appropriate

**Lab Lecture 3**

**SQL Arithmetic Operators** – used in expressions that evaluate to numeric values

**\*** multiplication

**/** division

**%** modulo, remainder

**+** addition

**-** subtraction

**DIV** (integer division), **MOD** (modulo), **-** (unary negative sign)

**Column1 \* Column2**

**Quantity \* UnitPrice**

**Quantity \* UnitPrice – Discount**

**SQL Comparison Operators** – used in expressions that evaluate to true or false

**=** checks equality

**<=>** checks equality, allows **NULL** (can also use **IS NULL**)

**<>** checks not equal

**!=** checks not equal

**>** greater than

**>=** greater than or equal

**<** less than

**<=** less than or equal

**NOTE:** When filtering to select all rows that do not have a value (or range of values), rows with **NULL** will **not** be returned.

**Strings (Text and Dates)** – single quotes needed (straight quotes), not case sensitive by default when performing comparison

* **Column1 = 'Text'**
* **Department = 'Camping'**
* **DateOnWebsite <> '14-05-21'**

**Numbers** – quotes not needed

* **Column1 > Column 2**
* **SKU = 100100**
* **ExtendedPrice < 100.75**

**WHERE** – specifies which rows are displayed in the query result