**CS 31 Database Management Programming Lecture 6**

**3NF but not BCNF** – a case of overlapping candidate keys

**GRADE**

|  |  |  |  |
| --- | --- | --- | --- |
| **StudentID** | **StaffID** | **ClassCode** | **EnrollGrade** |
| 125 | 25 | 21334 | A |
| 125 | 20 | 32456 | C |
| 135 | 20 | 28458 | B |
| 144 | 25 | 27563 | C |
| 144 | 20 | 32456 | B |

GRADE with StudentID and StaffID as composite primary key

**GRADE**

|  |  |  |  |
| --- | --- | --- | --- |
| **StudentID** | **StaffID** | **ClassCode** | **EnrollGrade** |
| 125 | 25 | 21334 | A |
| 125 | 20 | 32456 | C |
| 135 | 20 | 28458 | B |
| 144 | 25 | 27563 | C |
| 144 | 20 | 32456 | B |

GRADE with StudentID and ClassCode as composite primary key

The determinants are:

(STU\_ID, STAFF\_ID) → (CLASS\_CODE, ENROLL\_GRADE)

(STU\_ID, CLASS\_CODE) → (STAFF\_ID, ENROLL\_GRADE)

CLASS\_CODE → STAFF\_ID

**GRADE**

|  |  |  |
| --- | --- | --- |
| **StudentID** | **ClassCode** | **EnrollGrade** |
| 125 | 21334 | A |
| 125 | 32456 | C |
| 135 | 28458 | B |
| 144 | 27563 | C |
| 144 | 32456 | B |

**CLASS**

|  |  |
| --- | --- |
| **ClassCode** | **StaffID** |
| 21334 | 25 |
| 28458 | 20 |
| 27563 | 25 |
| 32456 | 20 |

Relations are now in BCNF

**Logical Conclusions**

* 2 column table and it has a single column primary key – BCNF
* 3 or greater column table and a single column primary key, no partial or transitive dependencies - BCNF
* 3 or greater column table and a composite primary key, no partial or transitive dependencies - 3NF

**Beyond BCNF**

**Multivalued Dependencies** – occur when a determinant is matched with a particular set of values

The notation for multivalued dependencies is →→.

**PARTKIT\_PRICE**

|  |  |  |
| --- | --- | --- |
| **PartKitName** | **Part** | **PartKitPrice** |
| Bike Repair | Screwdriver | 14.95 |
| Bike Repair | Tube Fix | 14.95 |
| Bike Repair | Wrench | 14.95 |
| First Aid | Aspirin | 24.95 |
| First Aid | Band Aids | 24.95 |
| First Aid | Elastic Band | 24.95 |
| First Aid | Ibuprofen | 24.95 |
| Toolbox | Drill | 74.95 |
| Toolbox | Drill Bits | 74.95 |
| Toolbox | Hammer | 74.95 |
| Toolbox | Saw | 74.95 |
| Toolbox | Screwdriver | 74.95 |

PARTKIT\_PRICE has multivalued dependencies

PartKitName →→ Part

PartKitName → PartKitPrice

**PRICE**

|  |  |
| --- | --- |
| PartKitName | PartKitPrice |
| Bike Repair | 14.95 |
| First Aid | 24.95 |
| Toolbox | 74.95 |

**PARTKIT**

|  |  |
| --- | --- |
| PartKitName | Part |
| Bike Repair | Screwdriver |
| Bike Repair | Tube Fix |
| Bike Repair | Wrench |
| First Aid | Aspirin |
| First Aid | Band Aids |
| First Aid | Elastic Band |
| First Aid | Ibuprofen |
| Toolbox | Drill |
| Toolbox | Drill Bits |
| Toolbox | Hammer |
| Toolbox | Saw |
| Toolbox | Screwdriver |

Multivalued dependencies have been removed

**Fourth Normal Form (4NF)** – a relation in BCNF in which there are no multivalued dependencies or in which all attributes participate in a single multivalued dependency

**Fifth Normal Form (5NF)** – A normal form necessary to eliminate an anomaly where a table can be split apart but not correctly joined back together. Also known as Project-Join normal Form (PJ/NF).

**Domain/Key Normal Form (DK/NF)** – a relation in which all constraints are logical consequences of domains and keys

Should we always normalize? How much?

* If data will be read-only, denormalize. We have no issues with deletion, insertion, and update anomalies if the database is read-only. Usually occurs when moving data to data warehouse where it will be used for analysis.
* A large database that is heavily normalized will be harder to query and perform slower. Excessive joins are required. Sometimes BCNF is just "too pure."

**Lab Lecture 6**

**SQL Aggregate Functions**

**SUM(Column1)** – calculates sum on Column1 values

**AVG(Column2)** – calculates average of Column2 values (arithmetic mean)

**SUM** and **AVG** only work with numeric columns.

**COUNT(\*)** – count number of rows in relation

**COUNT(Column3)** – count number of rows where Column3 is not NULL

**MIN(Column4)** – calculates minimum of Column4

**MAX(Column5)** – calculates maximum of Column5

**COUNT**, **MIN**, and **MAX** work with all datatypes.

* You can use **DISTINCT** with all 5 aggregate functions above.
* You can alias column names using **AS** or by entering the name directly after the aggregate call.
* Table columns and aggregates should not be combined (see Query 2\_40). This is incorrectly combining dimensional data with scalar data. NOTE: This changes once we introduce the **GROUP BY** clause.
* Aggregates are not allowed in the **WHERE** clause.

**Calculated Columns**

SELECT (Quantity \* Price) AS ExtPrice FROM ORDER\_ITEM;

**Concatenation** – joining values together (by appending one to the other) to form a single value

**CONCAT('Hello ', 'World!') = 'Hello World!'**

**RTRIM** – trims white space on right

**LTRIM** – trims white space on left

**TRIM** – trims white space on both end