**Daniel Darcy - CS4430 Cheat Sheet**

- CREATE TABLE Product (

maker varchar(255),

model int,

type varchar(100),

primary key (maker, model)

);

//date - yyyy-mm-dd, price – decimal , time - hh:mm:ss //Other data types

- Find the ships sunk in the battle of the Denmark Strait.

πship( σbattle = ’Denmark Strair’ (σresult = ’sunk’ (Outcomes ))

- The treaty of Washington in 1921 prohibited capital ships heavier than 35,000 tons. List the ships that violated the treaty of Washington.

πname(σdisplacement > 35000 (Classes left\_outerjoin Ships))

- List the name, displacement, and number of guns of the ships engaged in the battle of Guadalcanal.

πship, displacement, guns(σbattle = ’Guadalcanal’ ( Classes \* (Ships\* Outcomes )))

- List customers (names) who've ordered books through an employee named 'Jones'.

πCname(Customers |><| Orders |><| σEname=’Jones’ (Employees))

- Attributes – Column Headers – ATOMIC value

- Tuples – Rows - LIST

- Relation Name – Table Name – SET

- Relation schema - Relation Name and Attribute List – beers(name: string, manf: string)

- Database – Collection of relations

- Database schema – Set of all relation schemas in the database

- Tuples cannot have the same value in all key attributes

- SQL is primarily a query language but also includes data definition component for describing database schemas

- No attribute of a PRIMARY KEY can ever be NULL in any tuple. But attributes declared UNIQUE may have NULL’s, and there may be several tuples with NULL

- Algebra is a mathematical system consisting of operands: variables or values from which new values can be constructed, symbols denoting procedures that construct new values from given values

- Selection – picking certain rows, σ, WHERE

- Projection – picking certain columns, π, SELECT

- Products and joins: compositions of relations

- R1 := σ*C* (R2) - *C* is a condition (as in “if” statements) that refers to attributes of R2.

* + R1 is all those tuples of R2 that satisfy *C*.

- Product - R3 := R1 Χ R2

* + Pair each tuple t1 of R1 with each tuple t2 of R2.
  + Concatenation t1t2 is a tuple of R3.
  + Schema of R3 is the attributes of R1 and then R2, in order.
  + But beware attribute *A* of the same name in R1 and R2: use R1.*A* and R2.*A*.

- Renaming - R1 := ρR1(A1,…,A*n*)(R2) makes R1 be a relation with attributes A1,…,A*n* and the same tuples as R2.

Simplified notation: R1(A1,…,A*n*) := R2.

- Theta-Join - R3 := R1 ⋈*C* R2

* + Take the product R1 Χ R2.
  + Then apply σ*C* to the result.
* As for σ, *C* can be any boolean-valued condition.

- Natural Join - R3 := R1 ⋈ R2 connects two relations by:

* + Equating attributes of the same name, and
  + Projecting out one copy of each pair of equated attributes.

- A *bag* (or *multiset* ) is like a set, but an element may appear more than once.

- A *constraint*  is a relationship among data elements that the DBMS is required to enforce.

*- Triggers*  are only executed when a specified condition occurs, e.g., insertion of a tuple.

- A *view* is a relation defined in terms of stored tables (called *base tables* ) and other views.

a) *Virtual* = not stored in the database; just a query for constructing the relation.

b) *Materialized* = actually constructed and stored.

*- Index*  = data structure used to speed access to tuples of a relation, given values of one or more attributes.

- Could be a hash table, but in a DBMS it is always a balanced search tree with giant nodes (a full disk page) called a *B-tree*.

- To grant privileges, say:

GRANT <list of privileges>

ON <relation or other object>

TO <list of authorization ID’s>;

- If you want the recipient(s) to be able to pass the privilege(s) to others add:

WITH GRANT OPTION

* select desired attributes, from one or more tables, where condition about tuples of the tables
* \* in the select clause stands for all attributes of this selection
* For the result to have different attribute names use AS <new name>
* AND, OR, and NOT are Boolean operators
* *Pattern* is a quoted string with % = “any string”; \_ = “any character.”
* A parenthesized SELECT-FROM-WHERE statement (*subquery* ) can be used as a value in a number of places, including FROM and WHERE clauses.
* <tuple> IN (<subquery>) is true if and only if the tuple is a member of the relation produced by the subquery.
* EXISTS(<subquery>) is true if and only if the subquery result is not empty.
* *x* = ANY(<subquery>) is a boolean condition that is true iff *x* equals at least one tuple in the subquery result.
* *x* <> ALL(<subquery>) is true iff for every tuple *t* in the relation, *x* is not equal to *t*.
* For union, intersection and difference duplicates are eliminated as the operation is applied.

TRANSACTIONS

- Unlike operating systems, which *support*  interaction of processes, a DMBS needs to keep processes from troublesome interactions.

- Transaction = process involving database queries and/or modification.

- ACID Transactions

- Atomic: Whole transaction or none is done.

- Consistent: Database constraints preserved.

- Isolated: It appears to the user as if only one process executes at a time.

- Durable: Effects of a process survive a crash.

- The SQL statement COMMIT causes a transaction to complete.

- The SQL statement ROLLBACK also causes the transaction to end by aborting.

- Within a transaction we can say: SET TRANSACTION ISOLATION LEVEL X, where X = serializable, repeatable read, read committed, and read uncommitted.

- (Serializable) If Sally = (max)(min) and Joe = (del)(ins) are each transactions, and Sally runs with isolation level SERIALIZABLE, then she will see the database either before or after Joe runs, but not in the middle.

- Your choice, e.g., run serializable, affects only how *you* see the database, not how others see it.

- Example: If Joe Runs serializable, but Sally doesn’t, then Sally might see no prices for Joe’s Bar.

- (Read Uncommitted) A transaction running under READ UNCOMMITTED can see data in the database, even if it was written by a transaction that has not committed (and may never).

- Example: If Sally runs under READ UNCOMMITTED, she could see a price 3.50 even if Joe later aborts.

- (Read-Committed Transactions) If Sally runs with isolation level READ COMMITTED, then she can see only committed data, but not necessarily the same data each time.

- Example: Under READ COMMITTED, the interleaving (max)(del)(ins)(min) is allowed, as long as Joe commits.

Sally sees MAX < MIN.

- (Repeatable Read Transactions) Requirement is like read committed, plus: if data is read again, then everything seen the first time will be seen the second time.

- But the second and subsequent reads may see *more* tuples as well.

- Suppose Sally runs under REPEATABLE READ, and the order of execution is (max)(del)(ins)(min).

* + (max) sees prices 2.50 and 3.00.
  + (min) can see 3.50, but must also see 2.50 and 3.00, because they were seen on the earlier read by (max).