Schema and SQL/PSM

Database Schema

A Database Schema captures:

- The represented concepts
- Their attributes
- The constraints and dependencies over all attributes

You now know everything needed to build good schemas:

- ER modeling as the basis for capturing concepts
- Functional dependencies for constraining the data
- Normal forms for removing redundancy and anomalies
- Algorithms for decomposing and deriving normalized tables

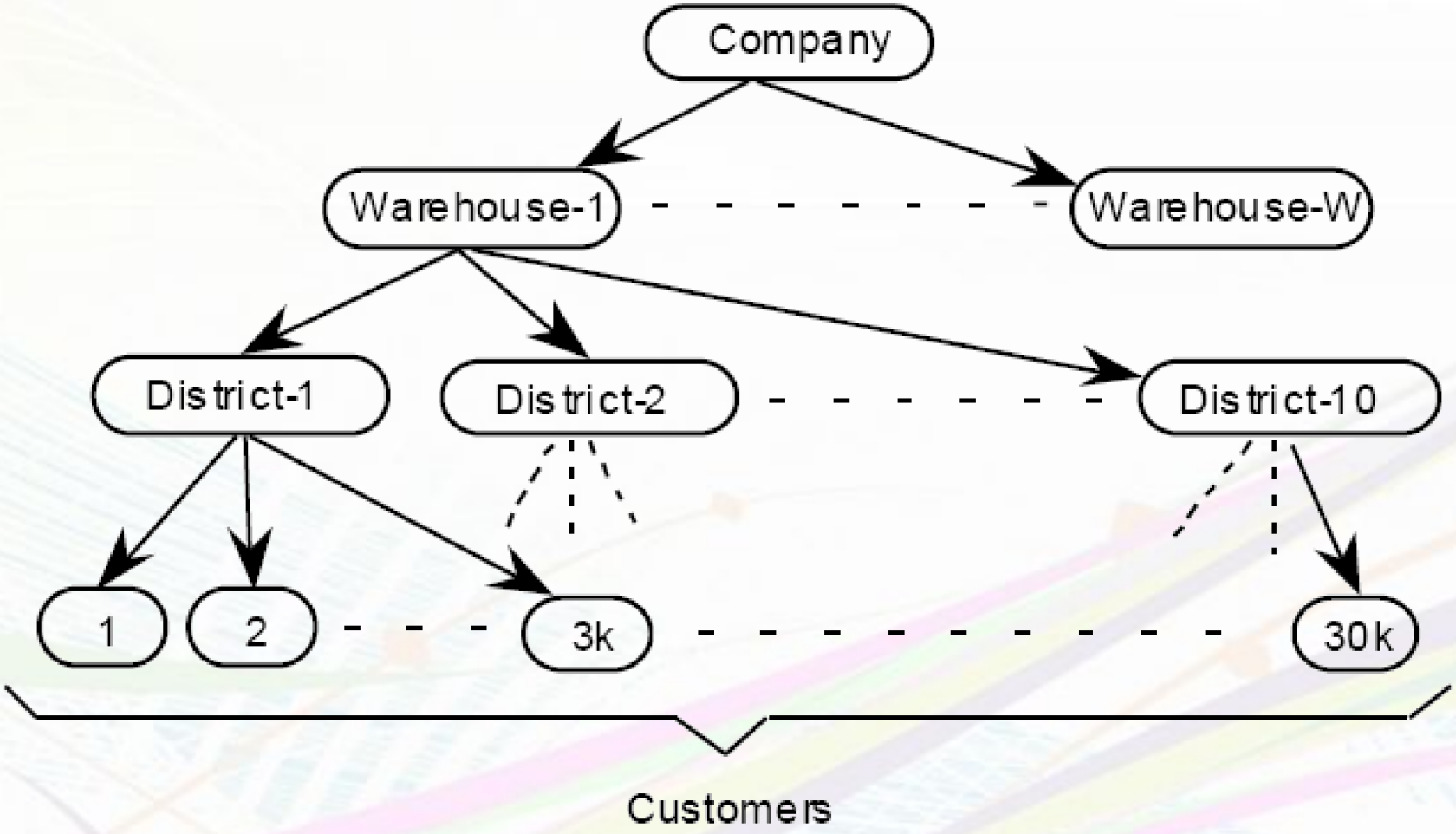
OLTP

- On-line Transaction Processing
- Refers to workloads with updates (UPDATE, INSERT, DELETE), often online (with response time requirements), and a high volume of, typically, small transactions (queries return few records, updates affect few records)
- Integrity is important, OLTP workloads are commonly run over 3NF schemas and many tables
- Examples: banking, e-shopping, sales, etc.

Example OLTP: TPC-C

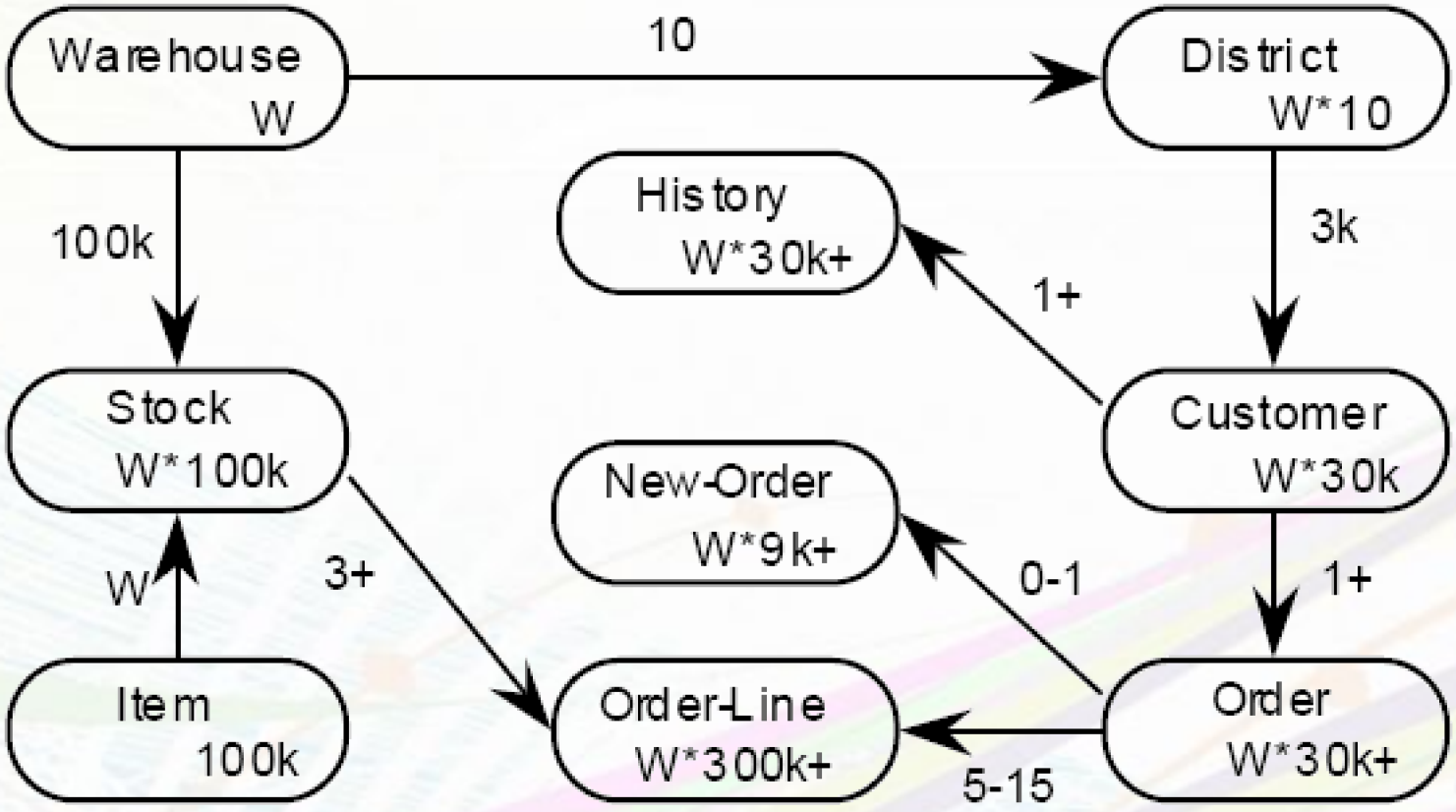
- http://www.tpc.org/tpcc/
- The Company portrayed by the benchmark is a wholesale supplier with a number of geographically distributed sales districts and associated warehouses. As the Company's business expands, new warehouses and associated sales districts are created. Each regional warehouse covers 10 districts. Each district serves 3,000 customers. All warehouses maintain stocks for the 100,000 items sold by the Company.

TPC-C Schema


Company
Warehouse-1
Warehouse-W
District-1
District-2
District-10
1
2
3k
30k
Customers

TPC-C Schema

- TPC-C has 9 tables



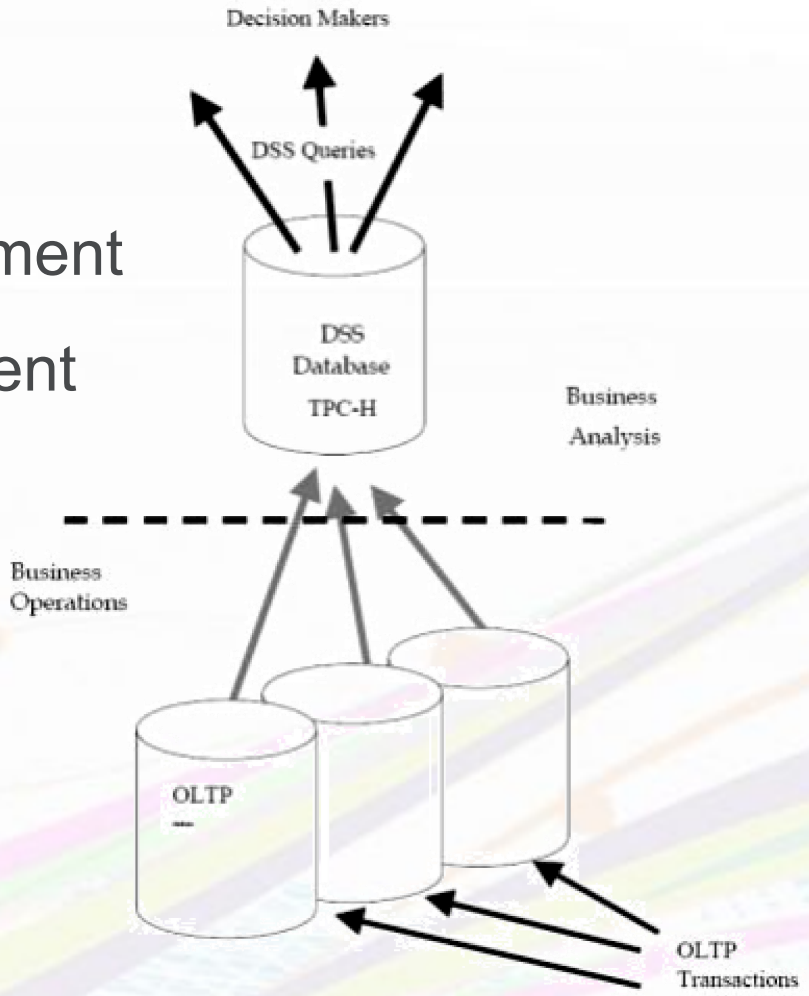
- http://www.tpc.org/tpcc/spec/tpcc_current.pdf

OLAP

- On-Line Analytical Processing
- Refers to workloads with heavy, complex queries (retrieving large number of records, often involving aggregation), data is updated but typically in batches (daily sales, weekly transactions, etc.), track changes over time.
- OLAP workloads commonly use de-normalized schemas (typically 2nf, flat and redundant), with some standard approaches such as star or snowflake schemas. For data mining, data cubes are used
- Examples: marketing analysis, reporting, data analysis

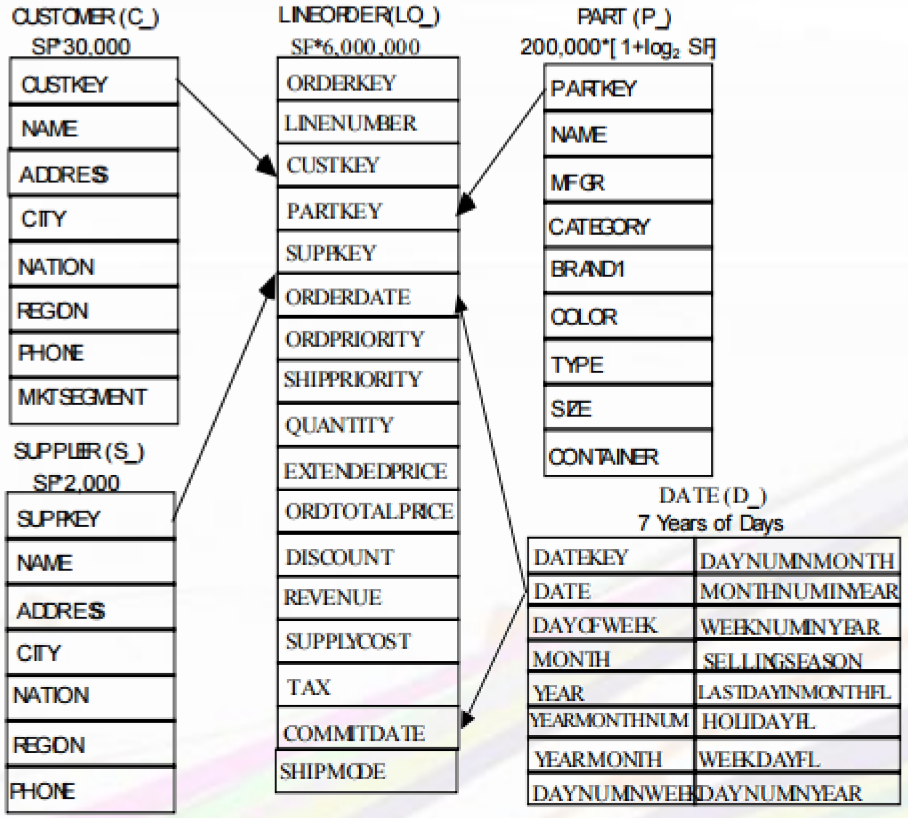
Example OLAP: TPC-H

- TPC-H queries address:
  - Pricing and promotions
  - Supply and demand management
  - Profit and revenue management
  - Customer satisfaction study
  - Market share study
  - Shipping management



Star schemas (SS benchmark)

- Star schemas are used in OLAP and data mining.
- Often referred as fact table + dimension tables
- Used when design centered around a very large collection of facts (sales, transactions, events, etc.)



http://www.tpc.org/tpctc/tpctc2009/tpctc2009-17.pdf
http://www.cs.umb.edu/~poneil/StarSchemaB.PDF

Example query (SS benchmark)

- The query is intended to provide revenue volume for lineorder transactions by customer nation and supplier nation and year within a given region, in a certain time period.

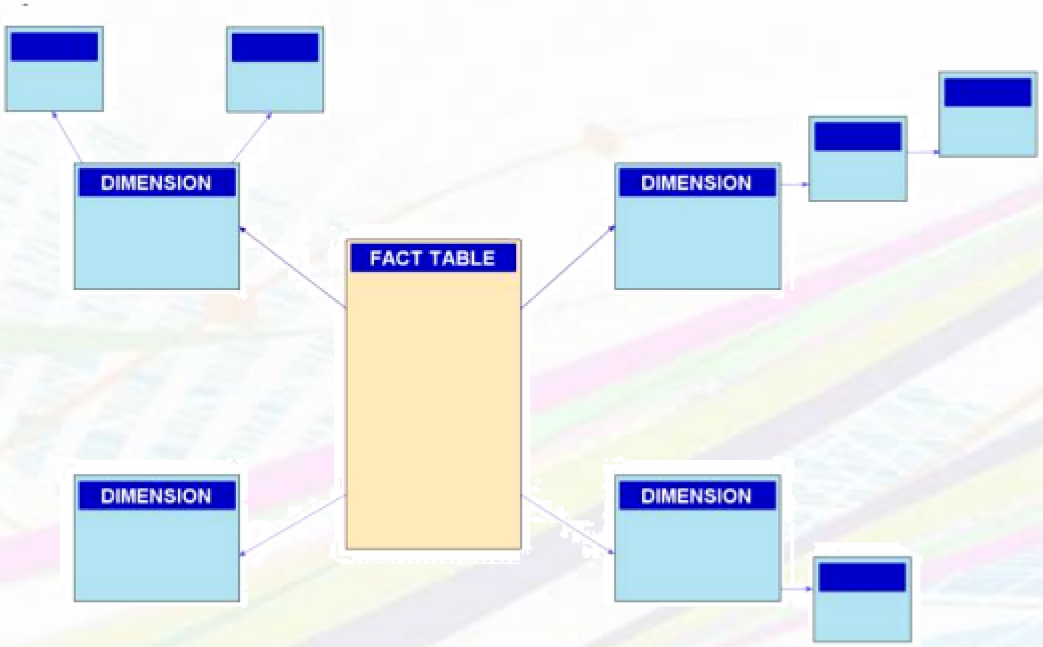
- select c_nation, s_nation, d_year, sum(lo_revenue) as revenue from customer, lineorder, supplier, date
  where lo_custkey = c_custkey
  and lo_suppkey = s_suppkey
  and lo_order date = d_datekey
  and c_region = 'ASIA'
  and s_region = 'ASIA'
  and d_year >= 1992
  and d_year <= 1997
  group by c_nation, s_nation, d_year
  order by d_year asc, revenue desc;

Normalization and schemas

- Normalization tries to avoid redundancy and anomalies. Why do de-normalized schemas work?
- OLAP databases are not usually populated by hand or individual transactions, data is loaded in batches and only periodically
- It is during the data loading process where the constraints and anomalies are controlled => responsibility is moved from the schema to the application doing the data loading
- This makes sense since the data loading can perform many transformations over the original data (see data cubes)

Snowflake schema

- In star schemas, the dimension tables and the fact table are not normalized
- A snowflake schema is a star schema where the dimension tables are normalized (some or all of them). Normalization is applied to low cardinality attributes to remove redundancy



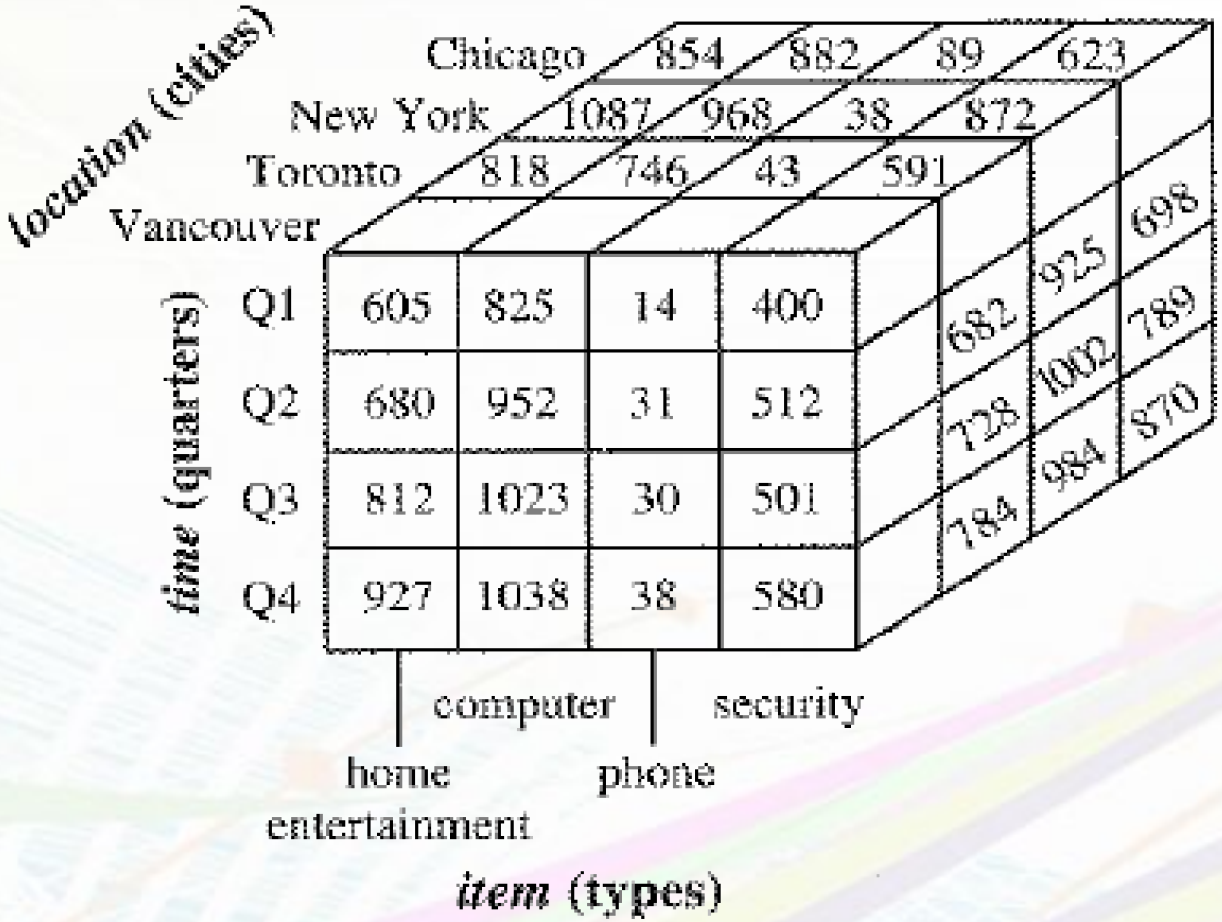
Modern trends

- Main memory databases (entire database or entire working set in main memory, no I/O for query processing)
- Column stores rather than row stores (at the physical design level not at the logical design level) -- http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf
- OLTP and OLAP in one system
- Heavy specialization for some applications (denormalization)
- Distributed databases, e.g., oceanbase alibaba, splice machine, etc.

Data cubes

- Data cubes are used for analysis and reporting.
- They include pre-aggregated data across several dimensions and granularities
- Support operations such as:
  - Slicing: selecting over one dimension (sales of a product by year and region)
  - Dicing: selection over the dimensions of the original cube (sales in June, July, August, from a range of products, in shops within a particular region)
  - Drill up and down: group-by at different granularities (sales by shop, sales in a shop by department, sales by product in a department)
  - Roll up: aggregation along a dimension (sales by week, month, year)

Data cubes examples



SQL/PSM

Procedures Stored in the Database

General-Purpose Programming

Stored Procedures

- An extension to SQL, called SQL/PSM, or "persistent, stored modules," allows us to store procedures as database schema elements.
- The programming style is a mixture of conventional statements (if, while, etc.) and SQL.
- Let's us do things we cannot do in SQL alone.

Basic PSM Form

CREATE PROCEDURE <name> (

<parameter list>)

<optional local declarations>

<body>;

- Function alternative:

CREATE FUNCTION <name> (

<parameter list>) RETURNS <type>

Parameters in PSM

- Unlike the usual name-type pairs in languages like C, PSM uses mode-name-type triples, where the *mode* can be:

  IN = procedure uses value, does not change value.

  OUT = procedure changes, does not use.

  INOUT = both.

Example: Stored Procedure

- Let's write a procedure that takes two arguments *b* and *p*, and adds a tuple to Sells that has bar = 'Joe''s Bar', beer = *b*, and price = *p*.

  Used by Joe to add to his menu more easily.

The Procedure

```
CREATE PROCEDURE JoeMenu (
    IN  b       CHAR(20),
    IN p        REAL
)
INSERT INTO Sells
VALUES('Joe''s Bar', b, p);
```

Parameters are both read-only, not changed

The body --- a single insertion

Invoking Procedures

- Use SQL/PSM statement CALL, with the name of the desired procedure and arguments.
- Example:

```
CALL JoeMenu('Moosedrool', 5.00);
```

- Functions used in SQL expressions where a value of their return type is appropriate.

Types of PSM statements -- 1

- RETURN <expression> sets the return value of a function.

  Unlike C, etc., RETURN *does not* terminate function execution.

- DECLARE <name> <type> used to declare local variables.

- BEGIN . . . END for groups of statements.

  Separate by semicolons.

Types of PSM Statements -- 2

- Assignment statements:

  SET <variable> = <expression>;

  Example: SET b = ’ Bud’ ;

- Statement labels: give a statement a label by prefixing a name and a colon.

IF statements

- Simplest form:

  IF <condition> THEN

  <statements(s)>

  END IF;

- Add ELSE <statement(s)> if desired, as

  IF . . . THEN . . . ELSE . . . END IF;

- Add additional cases by ELSEIF <statements(s)>:

IF … THEN … ELSEIF … ELSEIF … ELSE … END IF;

Example: IF

- Let's rate bars by how many customers they have, based on Frequents(drinker, bar).

  <100 customers: 'unpopular'.

  100-199 customers: 'average'.

  >= 200 customers: 'popular'.

- Function Rate(b) rates bar b.

Example: IF (continued)

```
CREATE FUNCTION Rate (IN b CHAR(20) )
        RETURNS CHAR(10)
        DECLARE cust INTEGER;
BEGIN
    SET cust = (SELECT COUNT(*) FROM Frequents
                WHERE bar = b);
    IF cust < 100 THEN RETURN 'unpopular'
    ELSEIF cust < 200 THEN RETURN 'average'
    ELSE RETURN 'popular'
    END IF;
END;
```

Number of customers of bar b

Nested IF statement

Loops

- Basic form:

  LOOP <statements> END LOOP;

- Exit from a loop by:

  LEAVE <loop name>

- The <loop name> is associated with a loop by prepending the name and a colon to the keyword LOOP.

Other Loop Forms

- WHILE <condition>

  DO <statements>

  END WHILE;

- REPEAT <statements>

  UNTIL <condition>

  END REPEAT;

Queries

- General SELECT-FROM-WHERE queries are *not* permitted in PSM.
- There are three ways to get the effect of a query:

1. Queries producing one value can be the expression in an assignment.
2. Single-row SELECT . . . INTO.
3. Cursors.

Example: Assignment/Query

- If *p* is a local variable and Sells(bar, beer, price) the usual relation, we can get the price Joe charges for Bud by:

```
SET p = (SELECT price FROM Sells
    WHERE bar = 'Joe''s Bar' AND
        beer = 'Bud');
```

SELECT . . . INTO

- An equivalent way to get the value of a query that is guaranteed to return a single tuple is by placing INTO <variable> after the SELECT clause.
- Example:

```
SELECT price INTO p FROM Sells
WHERE bar = 'Joe"s Bar' AND
      beer = 'Bud';
```

Cursors

- A *cursor* is essentially a tuple-variable that ranges over all tuples in the result of some query.
- Declare a cursor *c* by:

DECLARE c CURSOR FOR <query>;

Opening and Closing Cursors

- To use cursor *c*, we must issue the command:

  OPEN c;

  The query of *c* is evaluated, and *c* is set to point to the first tuple of the result.

- When finished with *c*, issue command:

  CLOSE c;

Fetching Tuples From a Cursor

- To get the next tuple from cursor c, issue command:

  FETCH FROM c INTO x1, x2,…,x*n* ;

- The *x* ’s are a list of variables, one for each component of the tuples referred to by *c*.
- c is moved automatically to the next tuple.

Breaking Cursor Loops -- 1

- The usual way to use a cursor is to create a loop with a FETCH statement, and do something with each tuple fetched.
- A tricky point is how we get out of the loop when the cursor has no more tuples to deliver.

Breaking Cursor Loops -- 2

- Each SQL operation returns a *status*, which is a 5-digit number.

  For example, 00000 = "Everything OK," and 02000 = "Failed to find a tuple."

- In PSM, we can get the value of the status in a variable called SQLSTATE.

Breaking Cursor Loops -- 3

- We may declare a condition, which is a boolean variable that is true if and only if SQLSTATE has a particular value.
- Example: We can declare condition NotFound to represent 02000 by:

```
DECLARE NotFound CONDITION FOR
    SQLSTATE '02000';
```

Breaking Cursor Loops -- 4

- The structure of a cursor loop is thus:

```
cursorLoop: LOOP
    …
    FETCH c INTO … ;
    IF NotFound THEN LEAVE cursorLoop;
    END IF;
    …
END LOOP;
```

Example: Cursor

- Let ' s write a procedure that examines Sells(bar, beer, price), and raises by $1 the price of all beers at Joe's Bar that are under $3.

  Yes, we could write this as a simple UPDATE, but the details are instructive anyway.

The Needed Declarations

```
CREATE PROCEDURE JoeGouge( )
    DECLARE theBeer CHAR(20);
    DECLARE thePrice REAL;
    DECLARE NotFound CONDITION FOR
        SQLSTATE '02000';
    DECLARE c CURSOR FOR
        (SELECT beer, price FROM Sells
         WHERE bar = 'Joe''s Bar');
```

Used to hold beer-price pairs when fetching through cursor c

Returns Joe's menu

The Procedure Body

```
BEGIN
    OPEN c;
    menuLoop: LOOP
        FETCH c INTO theBeer, thePrice;
        IF NotFound THEN LEAVE menuLoop END IF;
        IF thePrice < 3.00 THEN
            UPDATE Sells SET price = thePrice+1.00
            WHERE bar = 'Joe''s Bar' AND beer = theBeer;
        END IF;
    END LOOP;
    CLOSE c;
END;
```

Check if the recent FETCH failed to get a tuple

If Joe charges less than $3 for the beer, raise it's price at Joe's Bar by $1.

From next week, we are going to teach Database internals, covering file organization, data storing, and indexing.

- For extended reading and reference (ch8-12):

  Raghu Ramakrishnan and Johannes Gehrke.

  Database Management Systems 3rd edition

Quiz

- Given R(A,B,C,D,E), FD's: {AB → E, D → C}, compute
  1. all superkeys
  2. key
  3. decompose R into BCNF

solutions

- 1. {A, B, D}, {A, B, C, D}, {A, B, D, E}, and {A, B, C, D, E} are all superkeys.
- 2. {A, B, D} is the only key
- 3. R1(A,B,E), R3(C, D), R4 (A, B, D)