DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

IBM DB2 Universal Database V8.1

Database Administration Certification Preparation Course

Maintained by Clara Liu

IBM Software Group

Objectives

- Recursive SQL
- Outer Join
- OLAP SQL
- CASE Expressions
- Typed Tables
- Materialized Query Tables

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

CASE Expressions

Typed Tables

Materialized Query Tables

IBM Software Group

Recursive SQL

- A recursive SQL statement is used when an SQL repeatedly uses the resulting set to determine further results.
- i.e. bill-of-materials or routing information

```
WITH path (origin, destiny, distance, stops)
  AS ( SELECT  f.origin,  f.destiny, f.distance
        FROM flights f
        WHERE  origin='Sweden'
      UNION ALL
        SELECT p.origin, f.destiny,
                  p.distance+f.distance, p.stops+1
        FROM flights f, path p
        WHERE p.destiny=f.origin )
SELECT origin, destiny, distance, stops FROM path
```

1

2

3

Recursive SQL - Result Sets

ORIGIN	DESTINY	DISTANCE	STOPS
---	---	---	---

SQL0347W The recursive common table expression 'DB2.PATH' can contain an infinite loop. SQLSTATE=01605

Sweden	New York	8000	0
Sweden	Chicago	8700	0
Sweden	Toronto	9000	1
Sweden	Chicago	10500	1
Sweden	Austin	11300	2

.

.

.

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

CASE Expressions

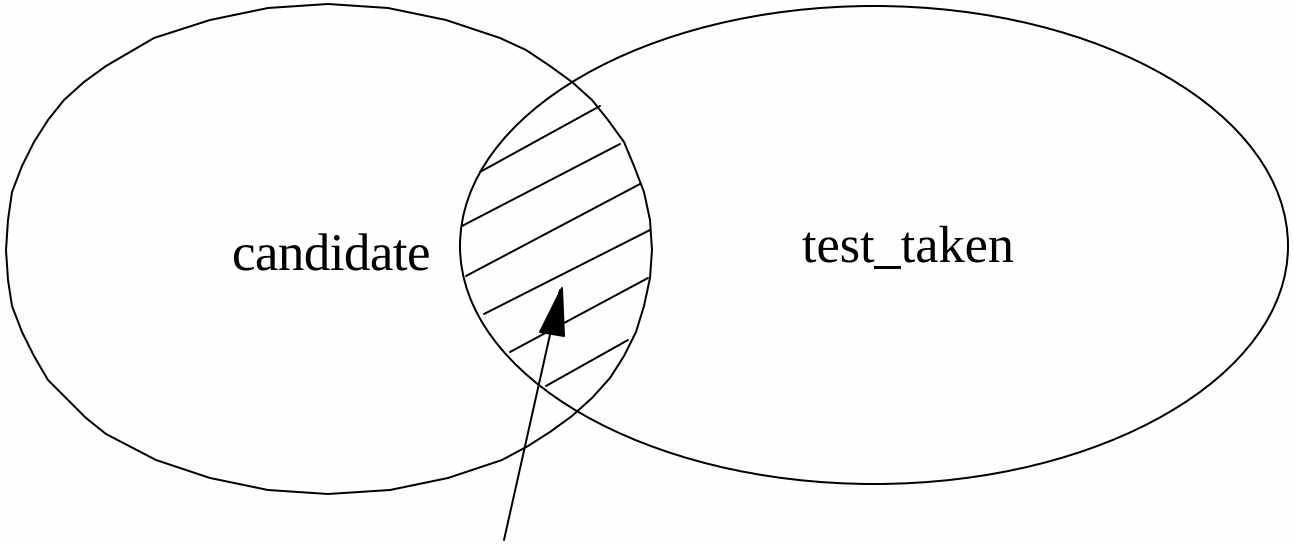
Typed Tables

Materialized Query Tables

IBM Software Group

Joins or Inner Joins

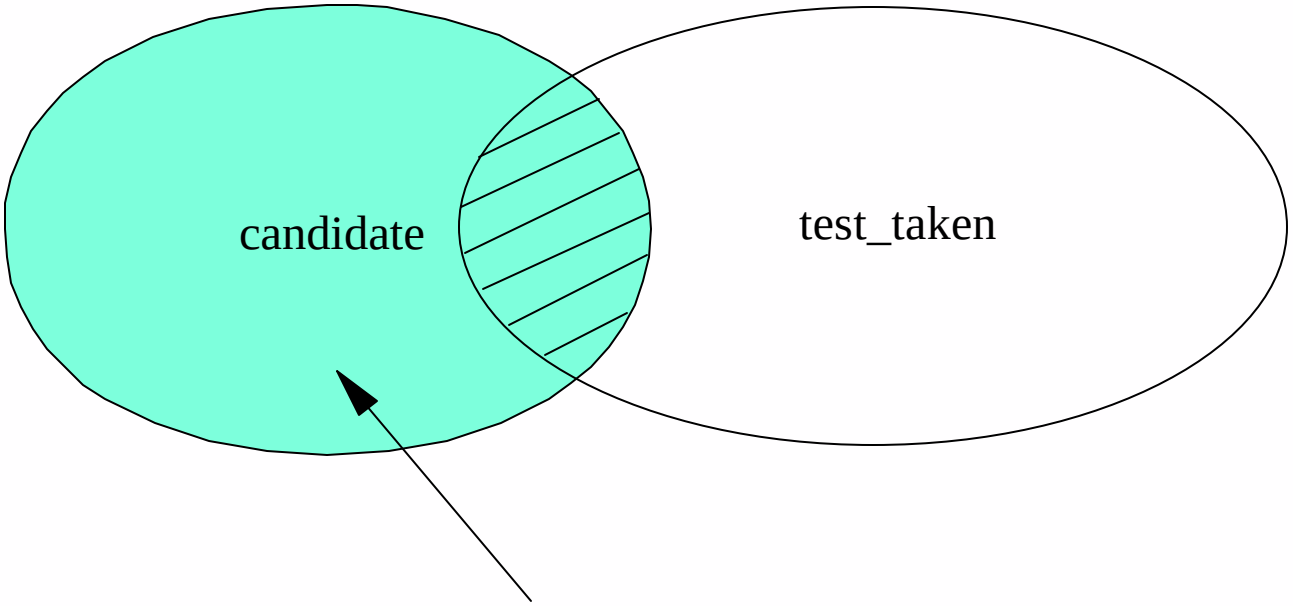
- Result set consists only of those matched rows that are present in both joined tables




```
SELECT fname, wphone, MAX(DECIMAL(score))
   FROM db2cert.candidate c
            INNER JOIN
            db2cert.test_taken tt ON c.cid=tt.cid
   GROUP BY fname, wphone
```

Left Outer Join

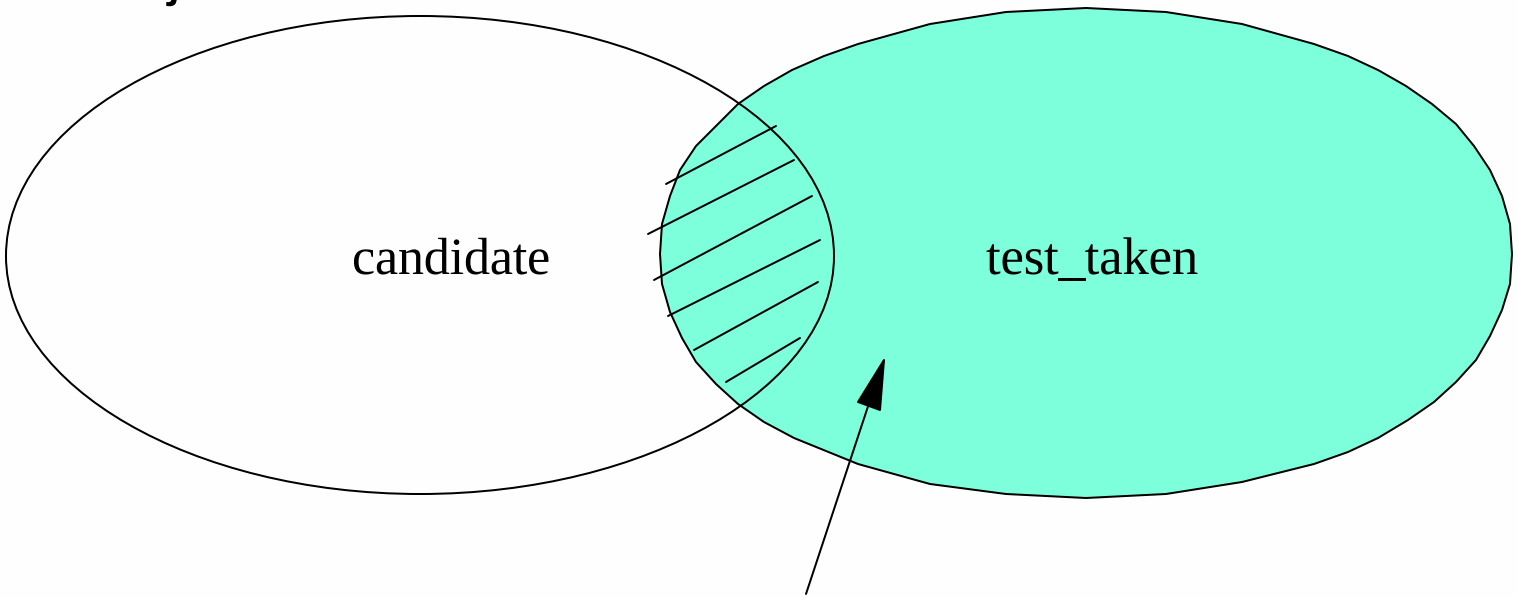
- Includes rows from the left table that were missing from the inner join




```
SELECT fname, wphone, MAX ( DECIMAL(score) )
   FROM db2cert.candidate c
            LEFT OUTER JOIN
            db2cert.test_taken tt ON c.cid=tt.cid
   GROUP BY fname, wphone
```

Right Outer Join

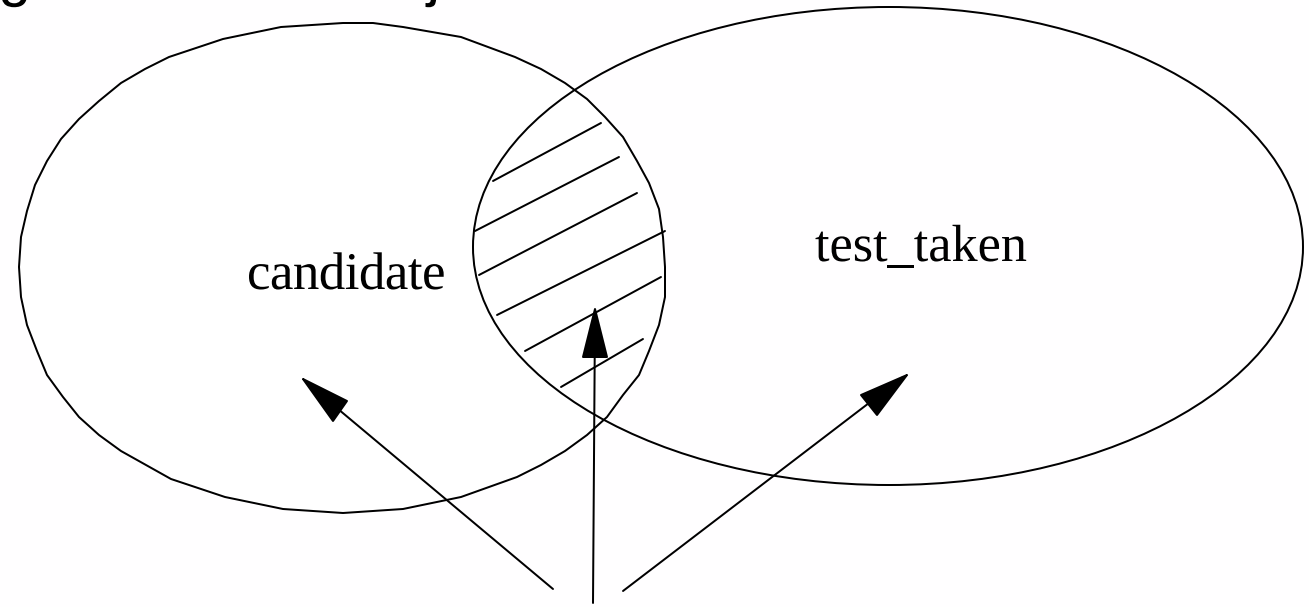
- Includes rows from the right table that were missing from the inner join




```
SELECT name, count ( DISTINCT char(tt.cid) )
   FROM db2cert.candidate c
            RIGHT OUTER JOIN
            db2cert.test_taken t ON c.number = t.number
  GROUP BY name
```

Full Outer Join

- Includes rows from both the left and right tables that were missing from the inner join




```
SELECT name, count ( DISTICT char(tt.cid) )
    FROM db2cert.test_taken tt
                FULL OUTER JOIN
                db2cert.test t ON tt.number = t.number
    GROUP BY name
```

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

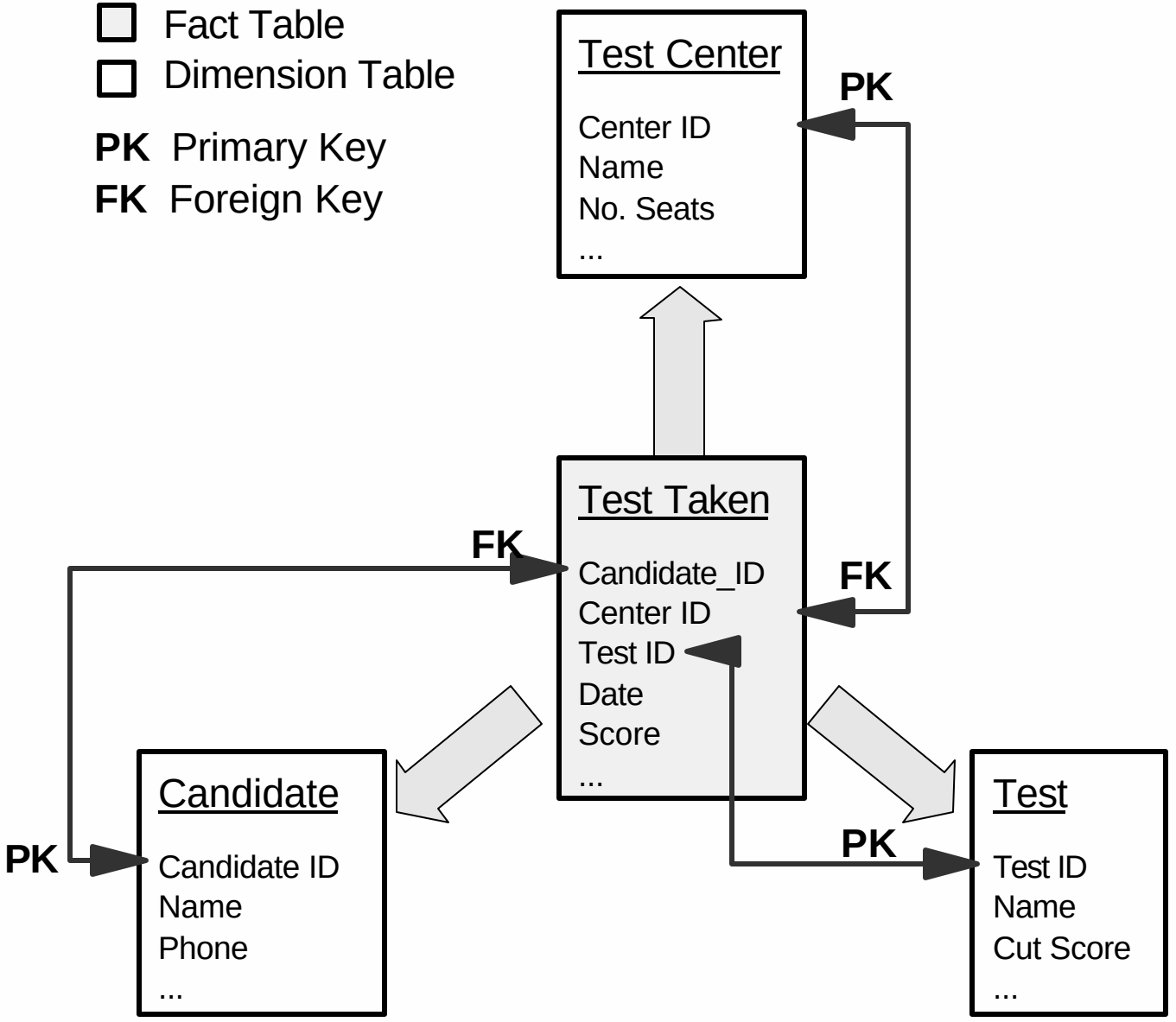
CASE Expressions

Typed Tables

Materialized Query Tables

IBM Software Group

Star Schema



Grouping Sets

- Allows multiple grouping clauses to be specified in a single statement
- Logically equivalent to the union of multiple subselects with the group by clause in each subselect corresponding to one grouping set
- Using grouping-sets allows the groups to be computed with a single pass over the base table

```
SELECT tt.tcid,t.name,count(*)

FROM db2cert.test_taken tt,
db2cert.test t

WHERE tt.number=t.number
GROUP BY GROUPING SETS
(tt.tcid,(tt.tcid,t.name))
```

TCID	NAME	3
TR01	-	**3**
TX01	-	**6**
TR01	**DB2 Application Development**	**2**
TR01	**DB2 Fundamentals**	**1**
TX01	**DB2 Administration**	**2**
TX01	**DB2 Application Development**	**2**
TX01	**DB2 Fundamentals**	**2**

Group By Rollup

- Extension to the GROUP BY clause that produces a result set that contains subtotal rows in addition to the "regular" grouped rows

```
SELECT c.country, tt.tcid, substr ( t.name,1,27 ) as test_name,
            count(*) as test_taken
  FROM db2cert.test_taken tt,db2cert.test t,db2cert.candidate c
  WHERE tt.number = t.number AND tt.cid = c.cid
  GROUP BY ROLLUP ( c.country,tt.tcid,t.name )
  ORDER BY c.country, tt.tcid, t.name
```

COUNTRY	TCID	TEST_NAME	TESTS_TAKEN
Canada	TX01	DB2 Administration	2
Canada	TX01	DB2 Application Development	2
Canada	TX01	DB2 Fundamentals	2
Canada	TX01	-	6
Canada	-	-	6
Germany	TR01	DB2 Application Development	2
Germany	TR01	DB2 Fundamentals	1
Germany	TR01	-	3
Germany	-	-	3
-	-	-	9

Group By Cube

- Produces a result set that contains all the rows of a ROLLUP aggregation and "cross-tabulation" rows

```
SELECT c.country, tt.tcid,
          SUBSTR(t.name,1,27) AS test_name,
          COUNT(*) AS tests_taken
FROM db2cert.test_taken tt,
        db2cert.test t, db2cert.candidate c
WHERE tt.number = t.number
   AND tt.cid = c.cid
GROUP BY
   CUBE (c.country,tt.tcid,t.name)
ORDER BY c.country,tt.tcid,t.name
```

COUNTRY	TCID	TEST_NAME	TESTS_TAKEN
Canada	TX01	DB2 Administration	2
Canada	TX01	DB2 Application Development	2
Canada	TX01	DB2 Fundamentals	2
Canada	TX01	-	6
Canada	-	DB2 Administration	2
Canada	-	DB2 Application Development	2
Canada	-	DB2 Fundamentals	2
Canada	-	-	6
Germany	TR01	DB2 Application Development	2
Germany	TR01	DB2 Fundamentals	1
Germany	TR01	-	3
Germany	-	DB2 Application Development	2
Germany	-	DB2 Fundamentals	1
Germany	-	-	3
-	TR01	DB2 Application Development	2
-	TR01	DB2 Fundamentals	1
-	TR01	-	3
-	TX01	DB2 Administration	2
-	TX01	DB2 Application Development	2
-	TX01	DB2 Fundamentals	2
-	TX01	-	6
-	-	DB2 Administration	2
-	-	DB2 Application Development	4
-	-	DB2 Fundamentals	2

Moving Function

- Allows column functions to be applied to a "window" of data

```
SELECT DAY,
   AVG ( SALES ) OVER
      ( ORDER BY DAY ROWS
         BETWEEN 1 PRECEDING AND 1 FOLLOWING )
    AS SMOOTH_VALUE FROM SALES ;
```

DAY	SMOOTH_VALUE
1	12
2	12
3	14
4	16
5	16
6	16
7	15
8	17
9	14
10	13

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

CASE Expressions

Typed Tables

Materialized Query Tables

IBM Software Group

CASE Expression

- Allow an expression to be selected based on the evaluation of one or more conditions
- If no case evaluates to true and the ELSE keyword is present then the result is the value of the result-expression or NULL
- If no case evaluates to true and the ELSE keyword is not present then the result is NULL
- CASE can be placed in SELECT clauses, WHERE predicates, grouping lists, functions, etc

```
SELECT COUNT (CASE WHEN decimal (score) > 90 then 1
                   ELSE null END) AS moregb90,
       COUNT (CASE WHEN decimal (score) = 90 then 1
                   ELSE null END) AS equalgb90,
       COUNT (CASE WHEN decimal (score) < 70 then 1
                   ELSE null END) AS minorgb70,
       COUNT (CASE WHEN number=test_id('500') then 1
                   ELSE null END) AS equalgb500
FROM db2cert.test_taken ;
```

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

CASE Expressions

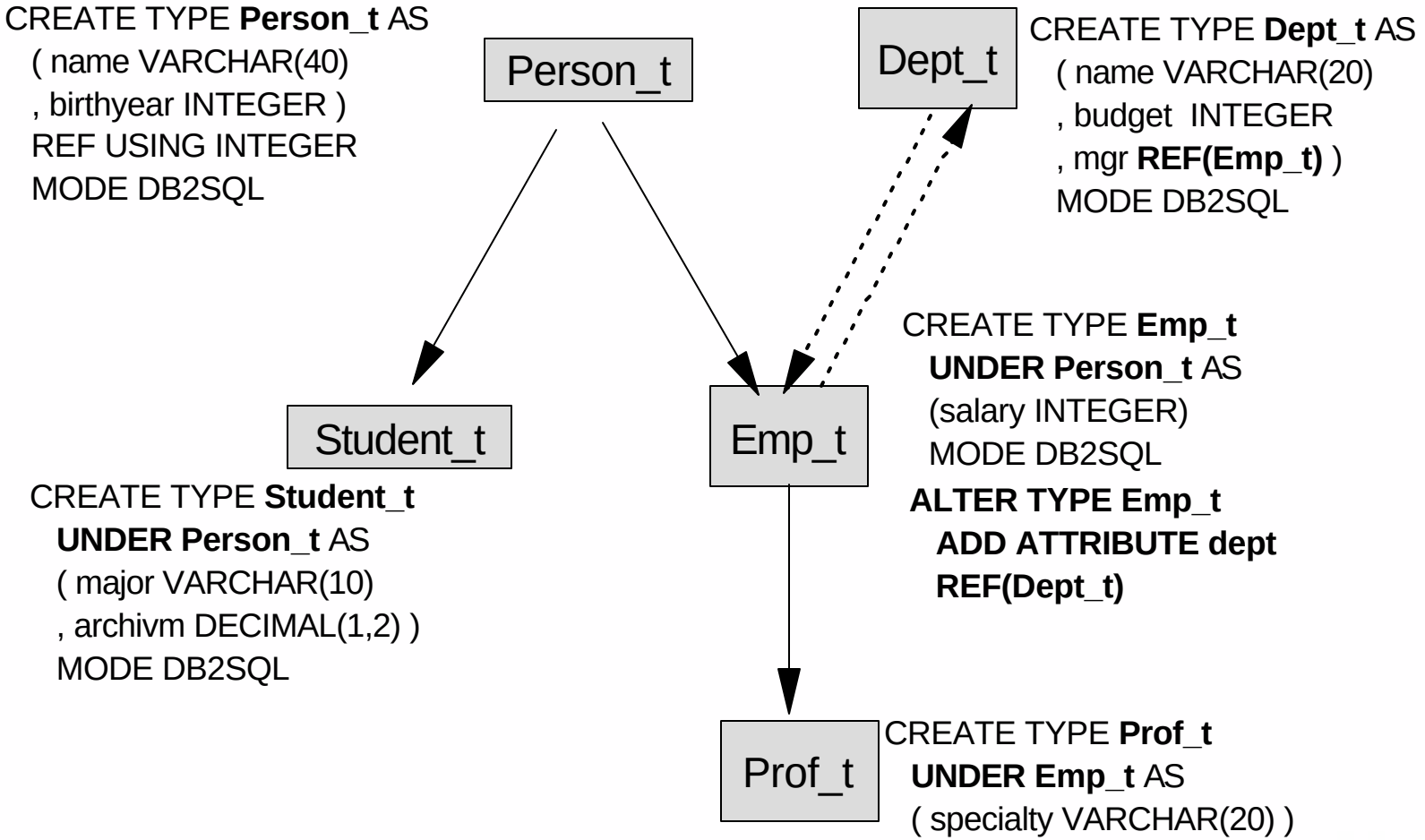
Typed Tables

Materialized Query Tables

IBM Software Group

Structured Types

- A user-defined structured type may include zero or more attributes
- May be a subtype allowing attributes to be inherited from a supertype



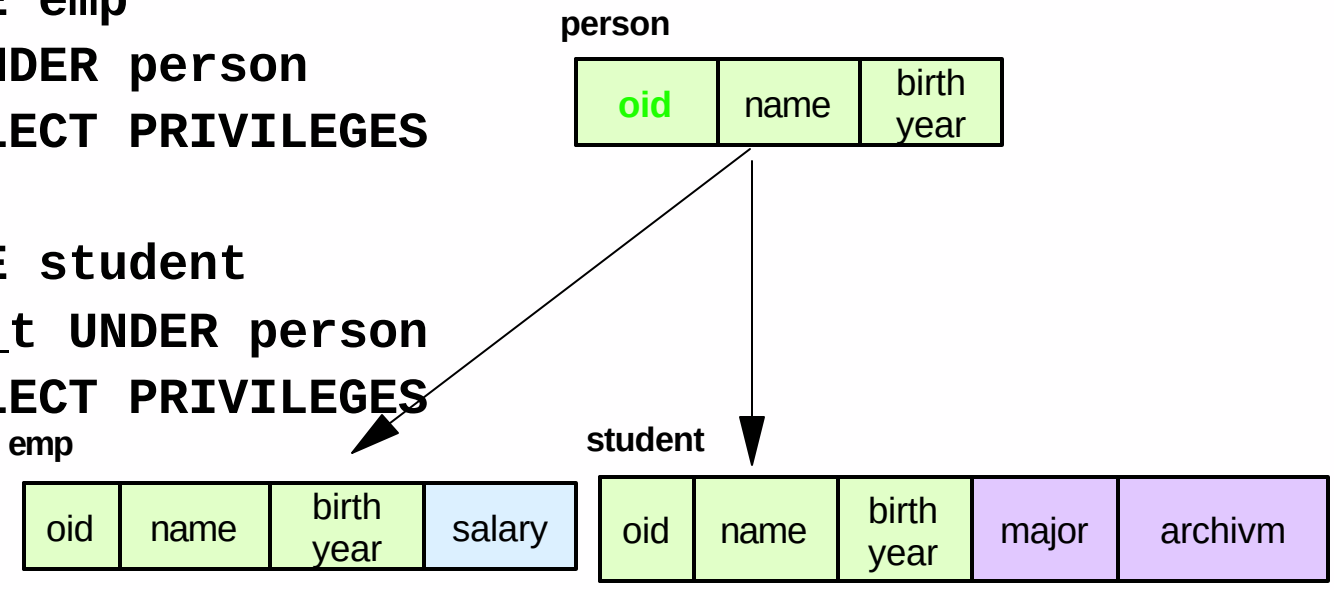
Typed Tables and Table Hierarchy

- Tables that contain structured types are called 'Typed Tables'
- Typed tables can inherit attributes from parent table or supertable
- Single inheritance only

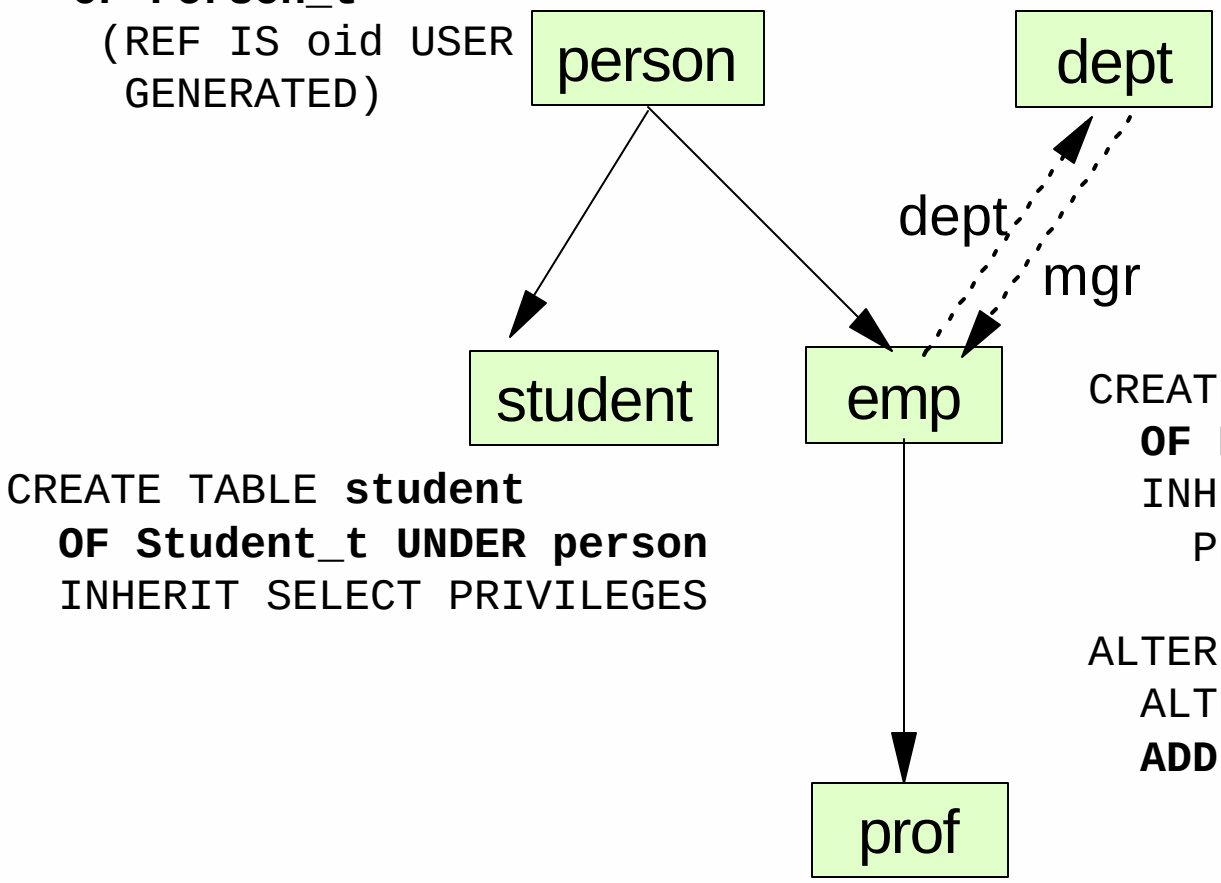
```
CREATE TABLE person
  OF Person_t
  (REF IS oid USER GENERATED)
```

```
CREATE TABLE emp
  OF Emp_t UNDER person
  INHERIT SELECT PRIVILEGES
```

```
CREATE TABLE student
  OF Student_t UNDER person
  INHERIT SELECT PRIVILEGES
```



Typed Tables - Example




```
CREATE TABLE person
 OF Person_t
  (REF IS oid USER
   GENERATED)
```

```
CREATE TABLE
  dept of Dept_t
  (REF IS oid
   USER GENERATED,
   mgr WITH OPTIONS
   SCOPE emp)
```

```
CREATE TABLE student
  OF Student_t UNDER person
  INHERIT SELECT PRIVILEGES
```

```
CREATE TABLE emp
  OF Emp_t UNDER person
  INHERIT SELECT
    PRIVILEGES

ALTER TABLE emp
  ALTER COLUMN dept
  ADD SCOPE dept
```

```
CREATE TABLE prof
  OF Prof_t UNDER emp
  INHERIT SELECT PRIVILEGES
```

Typed Tables - Example

Here is the content of all tables (assuming some rows are inserted).
All inherited columns are in **bold.**

Table Person

OID	NAME	BIRTHYEAR
10	John	1968
20	Paul	1961

Table dept

OID	NAME	BUDGET	MGR
1	math	300000	80
2	oec	500000	70
3	headq	5000000	90
4	itso	1000000	60

Table student

OID	**NAME**	**BIRTHYEAR**	MAJOR	ARCHIVM
100	Franzis	1975	pol	2,50
110	Herb	1980	math	1,70

Table emp

OID	**NAME**	**BIRTHYEAR**	SALARY	DEPT
30	Pat	1970	60000	1
40	Hitomi	1977	65000	2
90	Lou	-	-	-
50	Sam	1968	60000	4
60	Uta	1961	95000	3

Table prof

OID	**NAME**	**BIRTHYEAR**	**SALARY**	**DEPT**	SPECIALTY
70	Rich	1941	90000	3	oec
80	Herb	1962	120000	3	math

Hierarchy Table

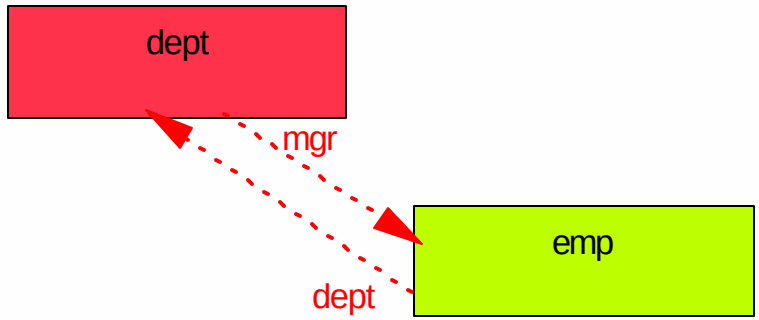
- Also known as the H-Table
- It holds all attributes of the tables in the table hierarchy
- There is one H-Table for each root type
- System catalog table SYSCAT.HIERARCHIES contains relationship between the sub-tables and super-tables
- The H-Table cannot be manipulated with SQL statements

Person_Hierarchy

TYPE_ID	OID	NAME	BIRTHYEAR	MAJOR	ARCHIVM	SALARY	DEPT	SPECIALTY
1035	10	John	1968	-	-	-	-	-
1037	100	Franzis	1975	pol	2.50	-	-	-
1039	30	Pat	1970	-	-	60000	1	-
1041	70	Rich	1962	-	-	120000	3	math

Reference Columns

- A column can be declared as a reference to another typed table (also called a target table)
- Value in reference column can identify a row exists in the target table or does not exist in the target table
- Similar, but not equal, to a foreign key
  - CREATE TABLE EMP (...
    DEPTNO REF(DEPT_TYPE) SCOPE DEPT,
    MGR REF(EMP_TYPE) SCOPE EMP, ...) ;
- A "dereference" operator X -> Y is introduced
- Meaning: follow reference X to its target table and select column Y
- Instead of a join we use this syntax:
  - SELECT name, salary, dept->name, dept->budget FROM emp
    WHERE dept->budget > 500000 ;



Example - SQL (1)

- **Insert an employee 'Tetsu' with oid '200', born 1968, $65000 a year and assign him to 'itso' department**

```
INSERT INTO emp (oid, name, birthday, salary, dept)
  VALUES (Empt_t(200), 'Tetsu', 1968, 65000,
      (SELECT oid FROM dept WHERE name='itso'))
```

- **Select all attributes of all employees (Emp_t and Prof_t) born after 1970 who earned more than $50000 a year:**

```
SELECT * FROM emp
WHERE birthyear > 1970 AND salary > 50000
```

- **Change the birthyear of a person (employee, professor, student) whose oid is 10 to be 1969:**

```
UPDATE person SET birthyear = 1969
WHERE oid = emp_t(10)
```

Example - SQL (2)

- **Find the name, salary, corresponding department and budget of employee working in departments with budget > 100000**

```
SELECT name, salary, dept->name, dept->budget
FROM emp
WHERE dept->budget > 500000
```

- **Find the name of all employees whose manager's manager is 'Lou'**

```
SELECT name
FROM emp
WHERE dept->mgr->dept->mgr->name = 'Lou'
```

Example - SQL (3)

- **Select employees who are inserted into table emp (No rows of subtables)**

```
SELECT * FROM only (emp)
WHERE dept->budget > 500000
```

- **Select all person who were born before 1965, and they are either student or person (excluding employee and professor)**

```
SELECT * FROM person
WHERE birthyear < 1965 AND
  DEREF(oid) is of dynamic type
(Student_t,only person_t)
```

DB2 Data Management Software

IBM

e business software

Chapter 7: Advanced SQL

Recursive SQL

Outer Join

OLAP SQL

CASE Expressions

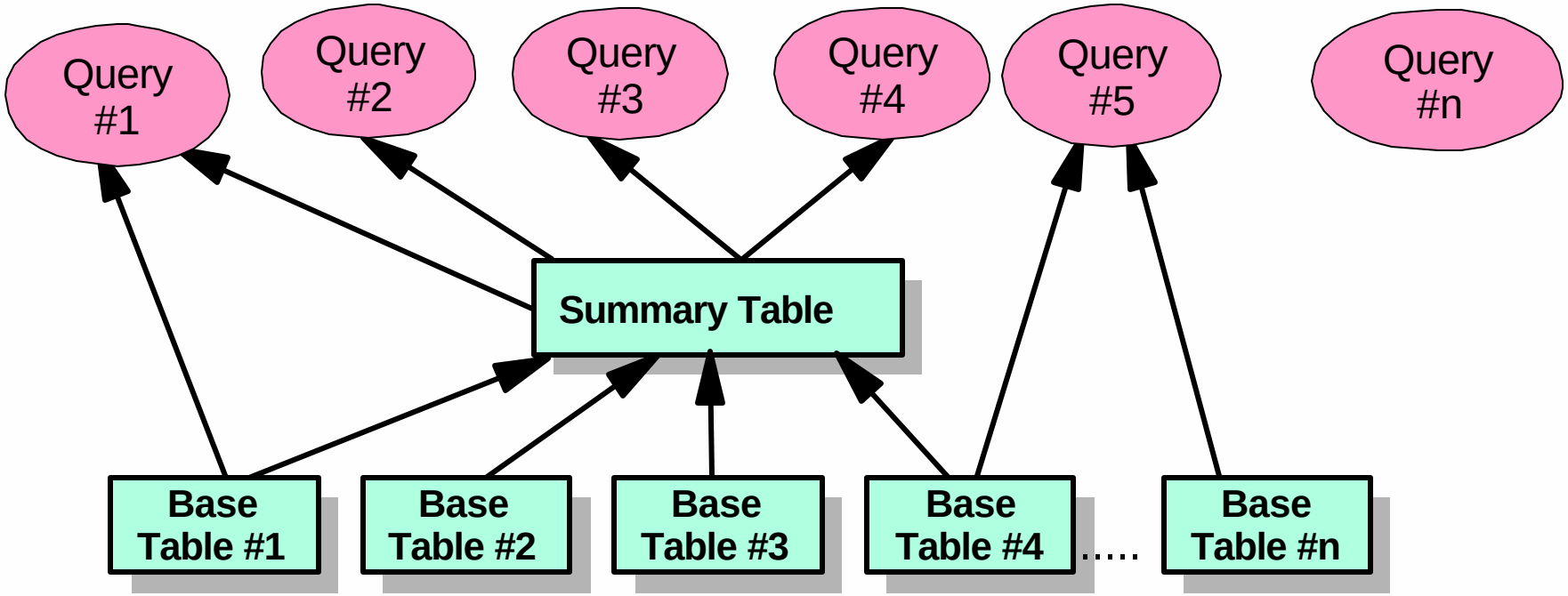
Typed Tables

Materialized Query Tables

IBM Software Group

Materialized Query Tables

- Called Summary Tables prior to DB2 V8
- Aggregate Aware Optimization - if the SQL compiler determines a query will run more efficiently against a materialized query table than the base table or tables, materialized query table will be used instead
- Definition based on the result of a query, contains precomputed results
- Improve performance and increase throughput of system



Materialized Query Tables

- Two types of materialized query tables:
  - MAINTAINED BY SYSTEM
    - Tables are mainted by the system
    - If base tables are updated, use the REFRESH option to indicate when the materialized query tables are refreshed:
      - REFRESH IMMEDIATE - when base data is changed, materialized query tables are refresh immediately
      - REFRESH DEFERRED - materialized query tables will not reflect changes to the underlying base table
  - MAINTAINED BY USER
    - Use custom applications to maintain and load the tables
    - Must be defined as REFRESH DEFERRED
- To manually refresh the materialized query table when the base table is changed, use the REFRESH TABLE statement
  - With activity affecting the source data, a materialized query table over time will no longer contain accurate data, use the REFRESH TABLE statement
- DATA INITIALLY DEFERRED option
  - Data is not inserted into the table as part of the CREATE TABLE statement
  - Use the REFRESH TABLE statement to populate data into the table

Materialized Query Tables - Examples

- Example:
  - CREATE TABLE abc (col1, col2, col3, col4)
    - AS (SELECT ..... FROM .....)
    - DATA INITIALLY DEFERRED
    - REFRESH IMMEDIATE
    - ENABLE QUERY OPTIMIZATION
    - MAINTAINED BY SYSTEM
- Example:
  - REFRESH TABLE abc INCREMENTAL
    - INCREMENTAL
      - Only refresh the appended portion content
      - If such a request cannot be satisfied, an error (SQLSTATE 55019) is returned
    - NOT INCREMENTAL
      - Specifies a full refresh for the table by recomputing the materialized query table definition

Use Materialized Query Tables For Query Optimization

- ENABLE QUERY OPTIMIZATION
  - Table can be used for query optimization under appropriate circumstances
- DISABLE QUERY OPTIMIZATION
  - Table will not be used for query optimization, it can still be queried directly
- Materialized query tables are never considered by static embedded SQL queries
- CURRENT REFRESH AGE special register
  - Specifies the amount of time that the materialized query table defined with REFRESH DEFERRED can be used for dynamic queries before it must be refreshed
  - To set CURRENT REFRESH AGE, use SET CURRENT REFRESH AGE statement
  - The CURRENT REFRESH AGE special register can be set to ANY, or 99999999999999, or a timestamp duration with a data type of DECIMAL(20,6)
  - A value of zero (0) indicates that only materialized query tables defined with REFRESH IMMEDIATE may be used to optimize the processing of a query
- CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register
  - Specifies a VARCHAR(254) value that identifies the types of tables that can be considered when optimizing dynamic SQL queries
  - SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION SYSTEM
  - SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION USER

Staging Tables

- A staging table allows incremental maintenance support for deferred materialized query table
- Collects changes that need to be applied to the materialized query table to synchronize it with the contents of underlying tables
- Eliminates the high lock contention caused by immediate maintenance content when an immediate refresh of the materialized query table is requested
- The materialized query tables no longer need to be entirely regenerated whenever a REFRESH TABLE is performed