DBA Certification Course (Summer 2008)

Chapter 4: Database Objects

- Data Types
- Integrity Constraints
- Generated Columns
- Temporary Tables

4-1



After completing this chapter, you should be able to:

- write CREATE TABLE statements in DB2
- use generated columns in DB2



 Hana Curtis: DB2 9 Fundamentals exam 730 prep, Part 5: Working with DB2 objects

[http://www.ibm.com/developerworks/edu/dm-dw-db2-cert7305.html]

 Clara Liu, Raul Chong, Dwaine Snow, Sylvia Qi: Understanding DB2:

Learning Visually with Examples

IBM Press/Pearson, 2005, ISBN 0-13-185916-1, 895 pages.

• DB2 for Linux, UNIX, and Windows Version 9 Information Center

[http://publib.boulder.ibm.com/infocenter/db2luw/v9//index.jsp]

Literature (2)

- Don Chamberlin:
 - A Complete Guide to DB2 Universal Database

Morgan Kaufmann, 1998, ISBN 1-55860-482-0, 795 pages.

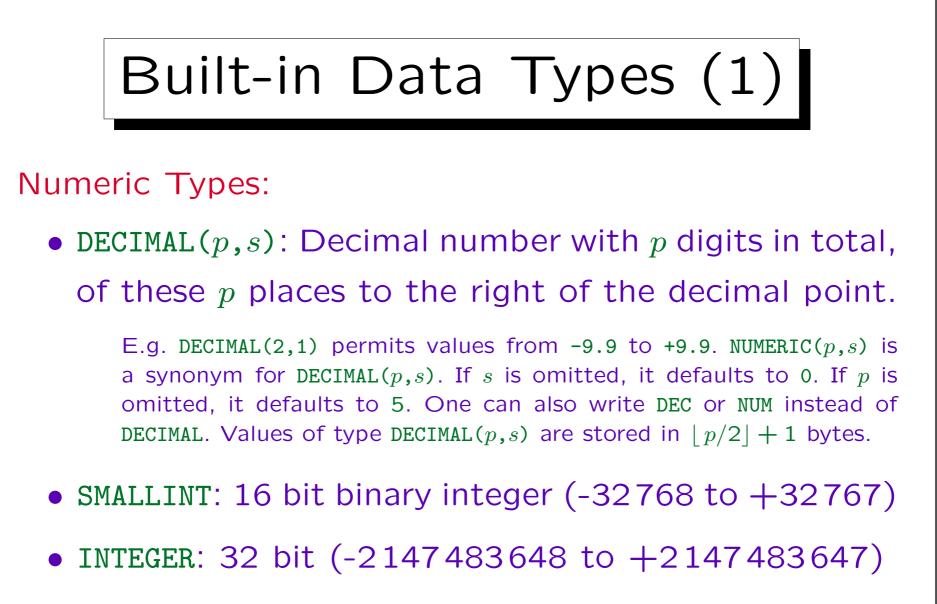
• Roger E. Sanders: DB2 Universal Database V8.1 Certification Exam 700 Study Guide

Prentice Hall / IBM Press, 2004, ISBN 0-13-142465-3, 416 pages.

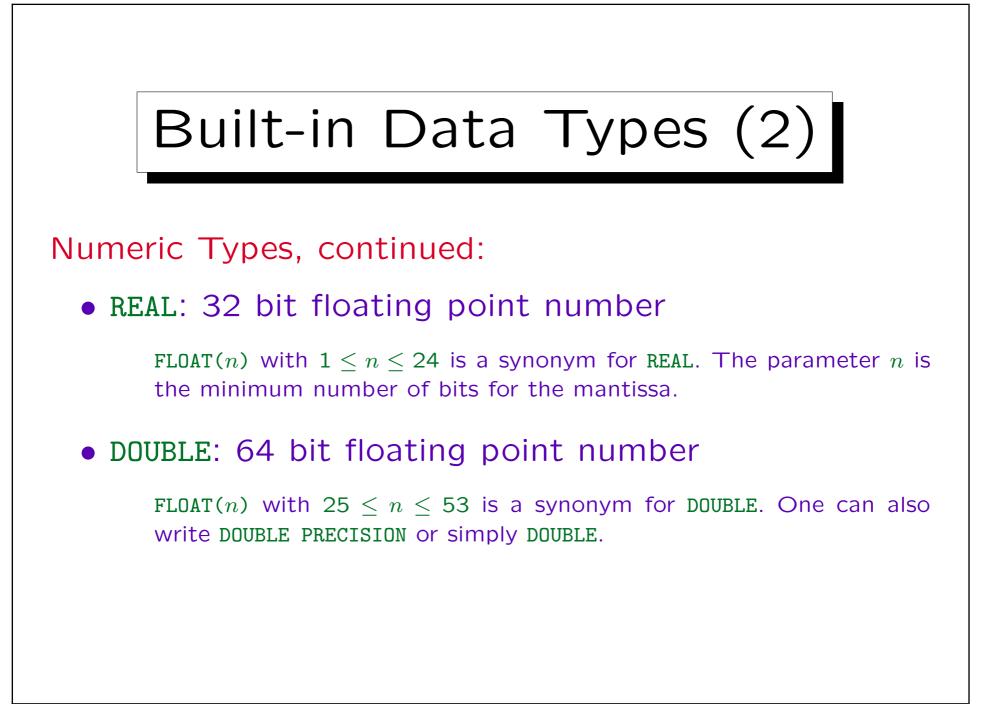


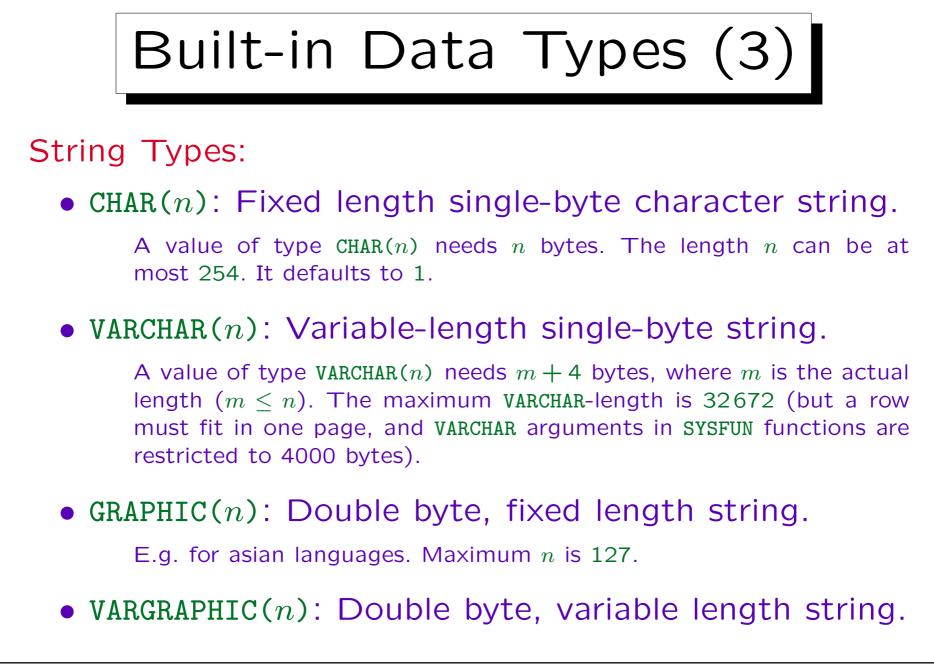


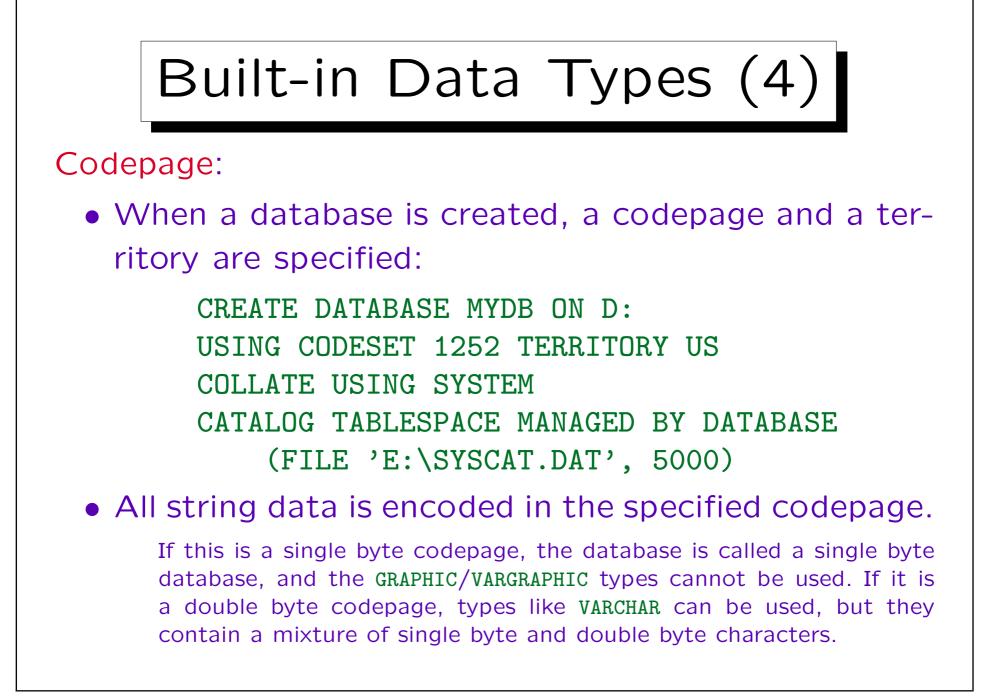
- 2. Tables
- 3. Indexes
- 4. Triggers
- 5. Typed Tables

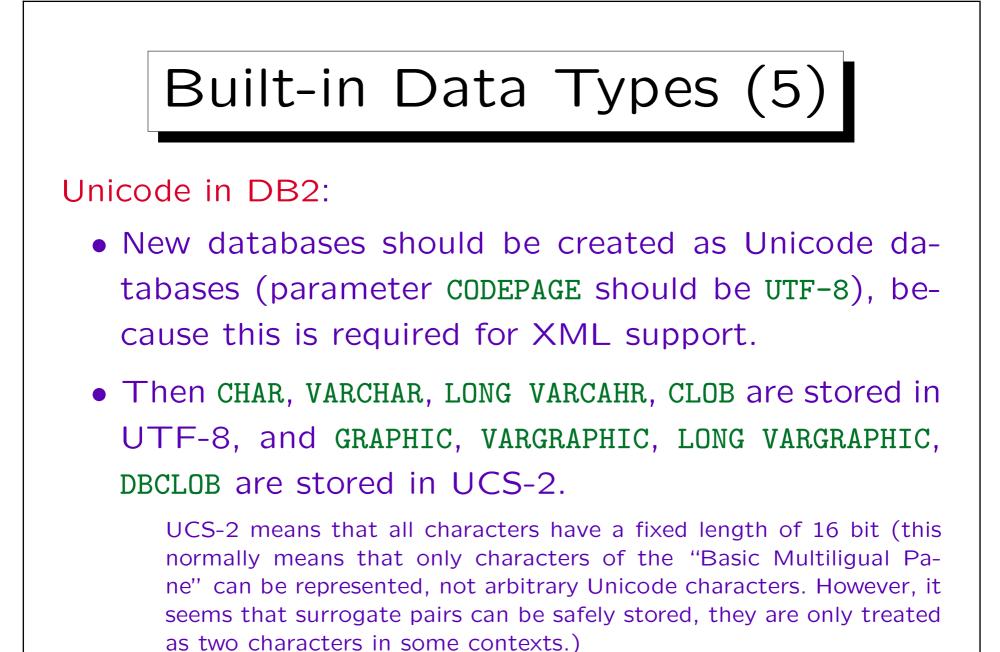


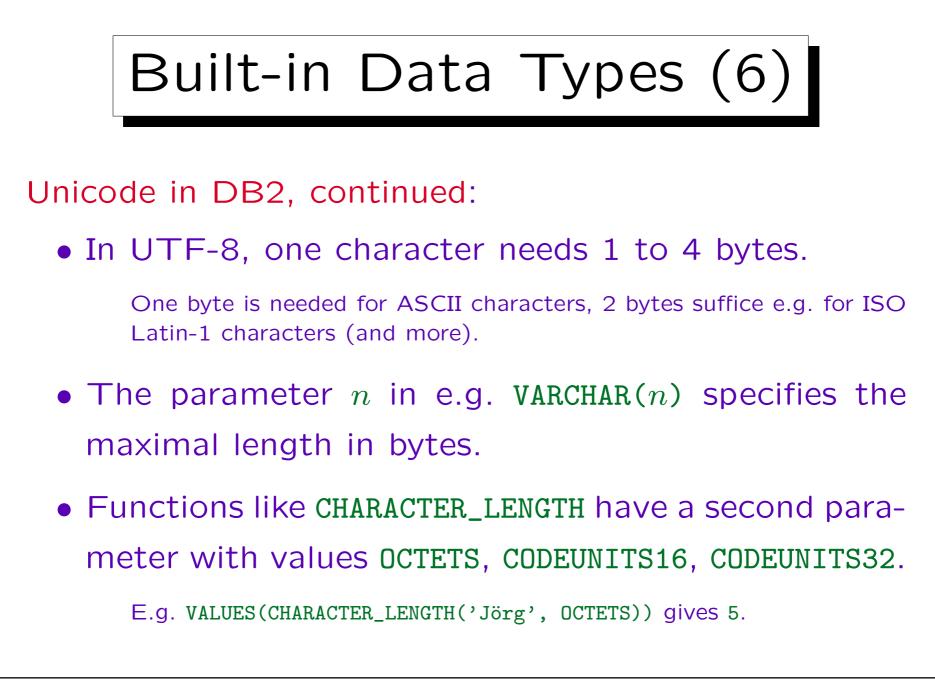
• BIGINT: 64 bit (more than 18 decimal places)

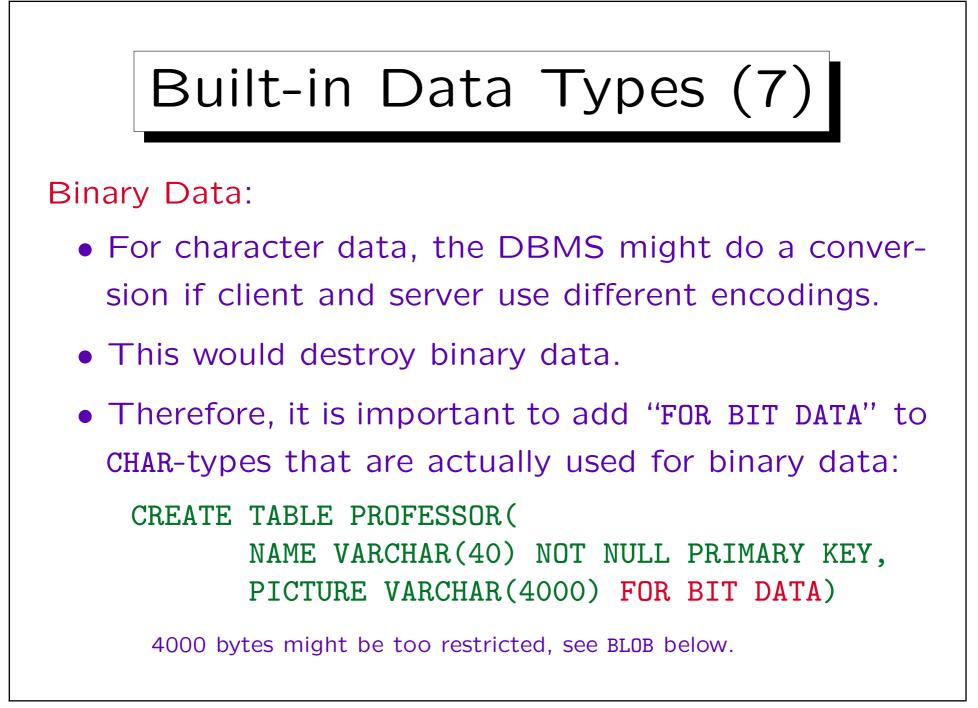


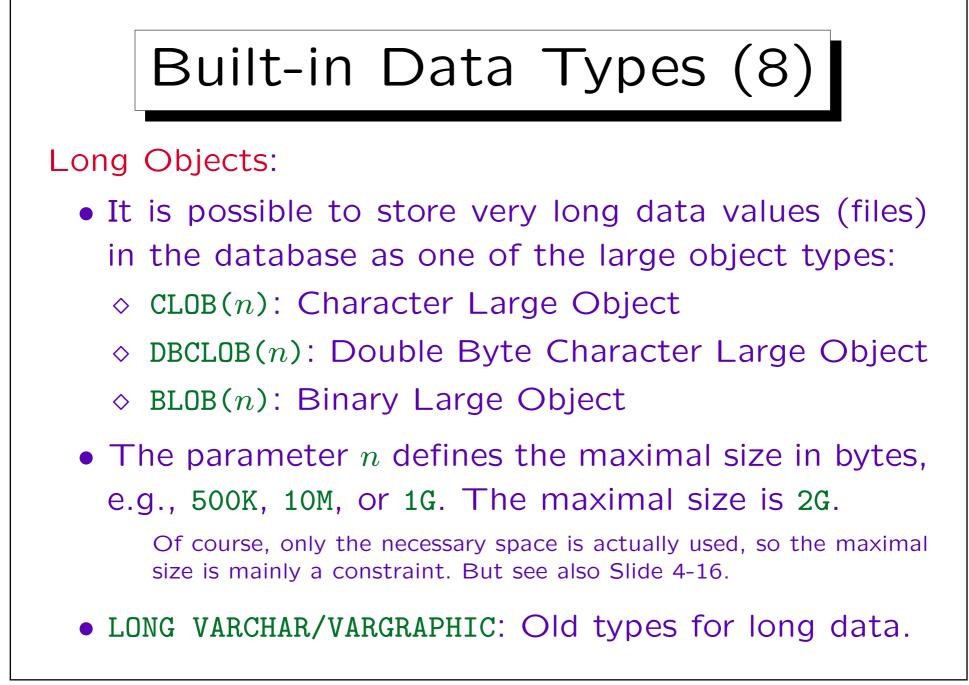


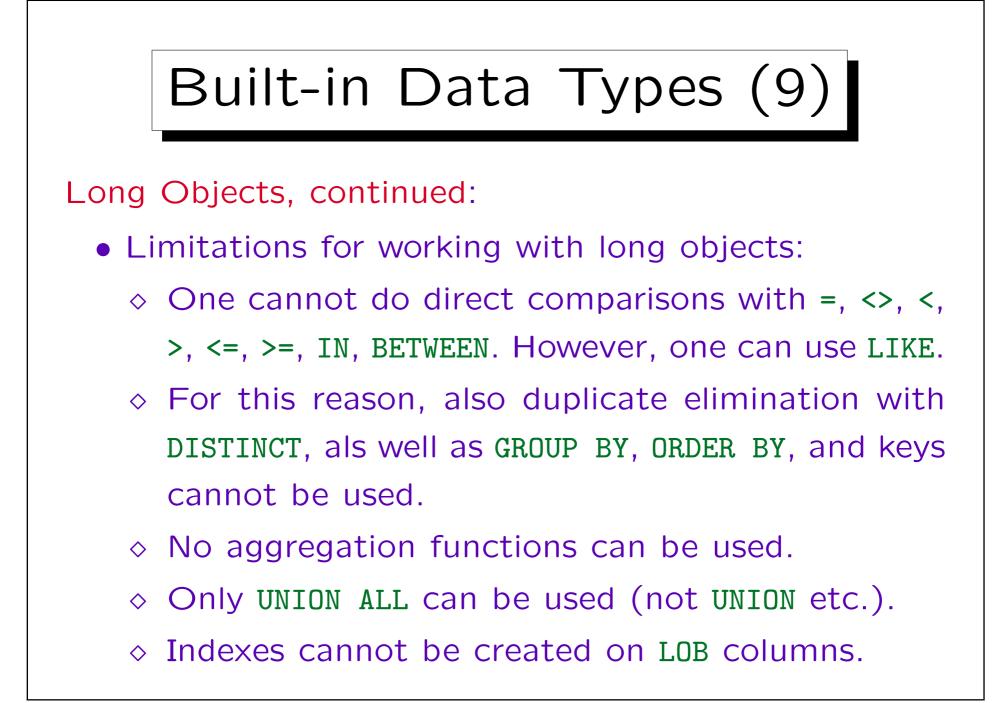


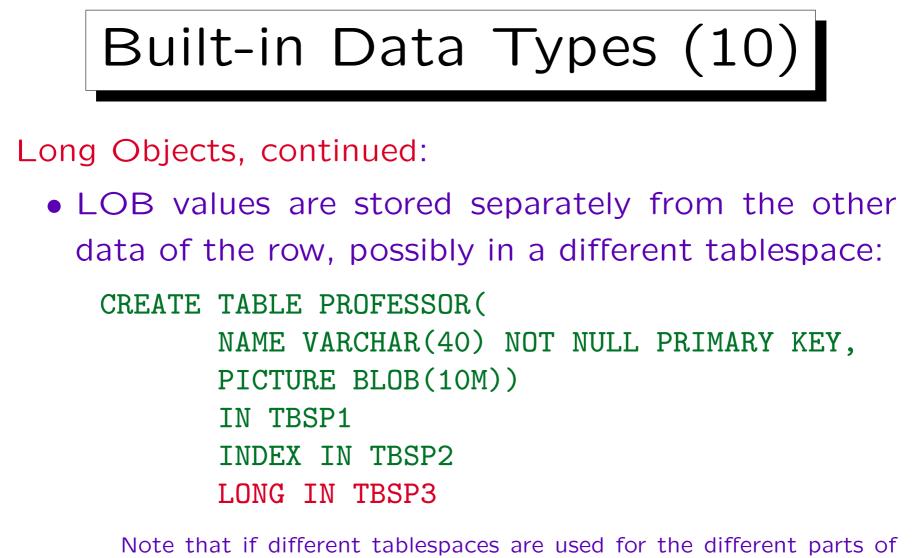




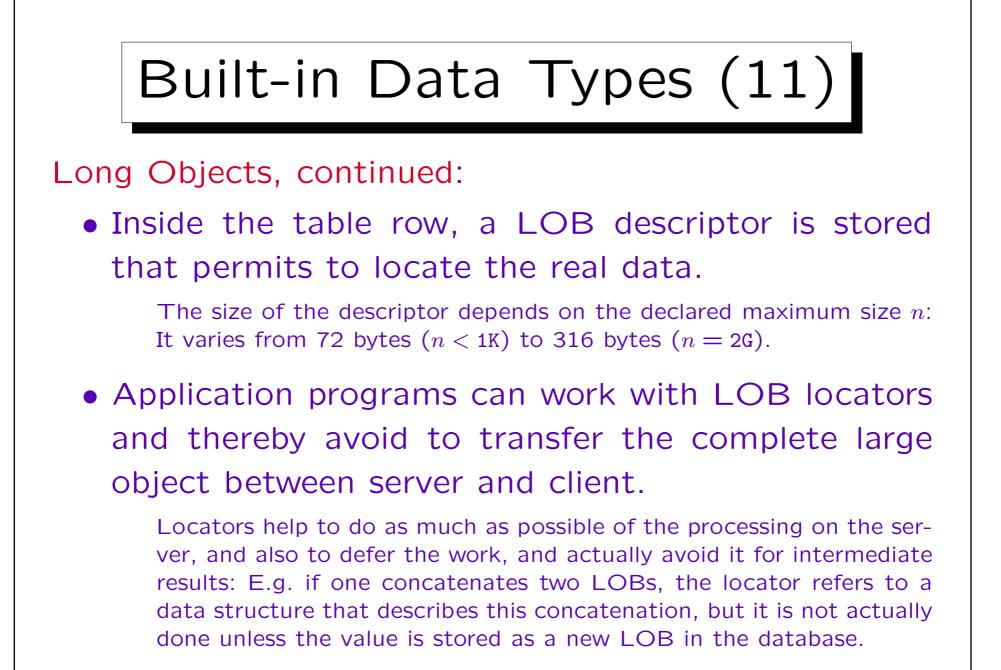


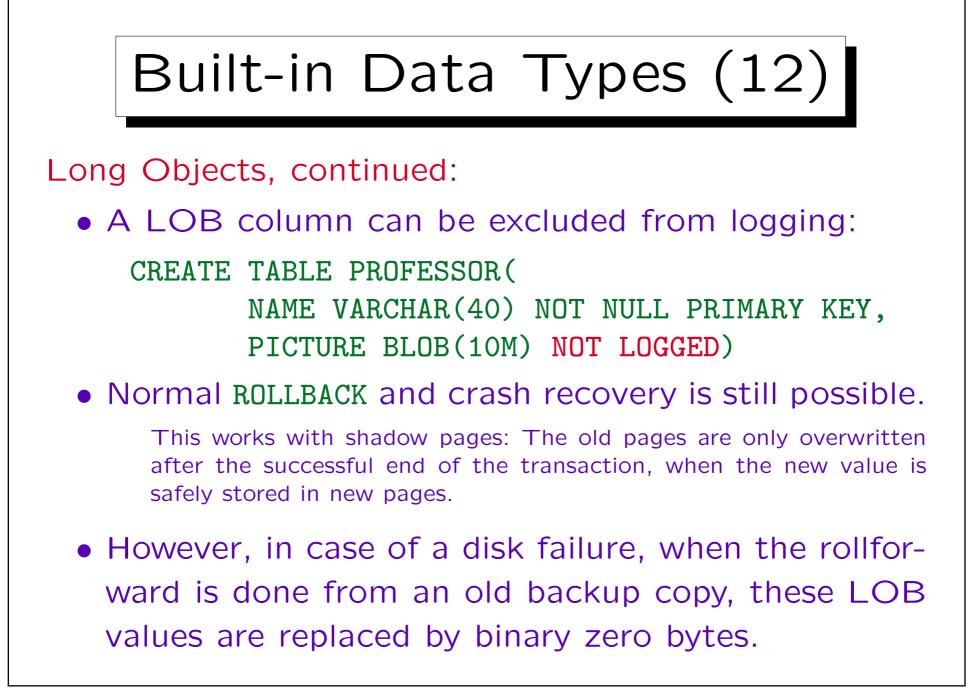


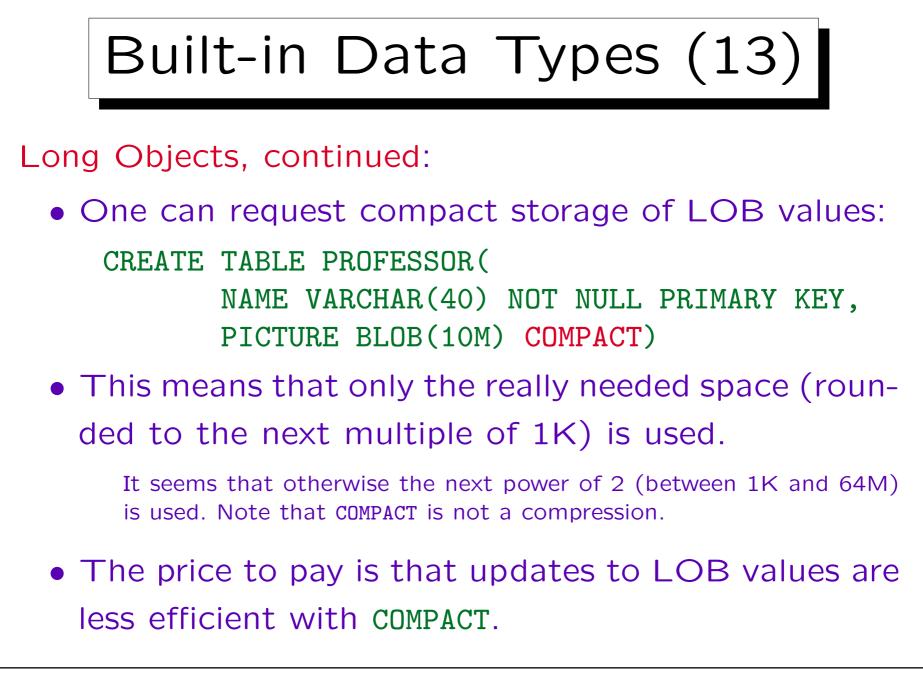


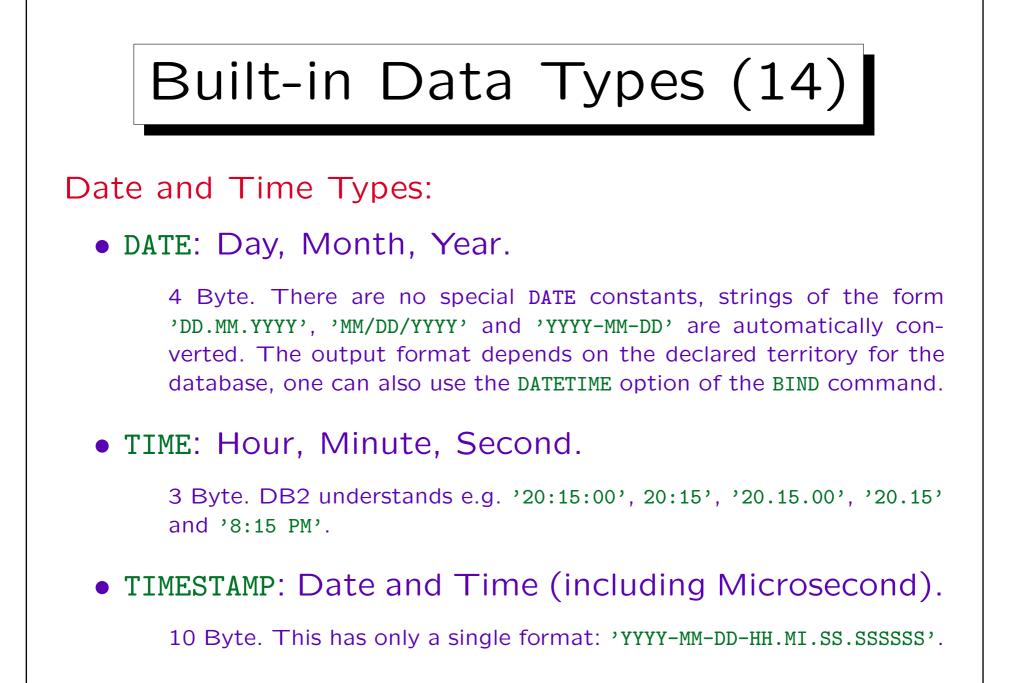


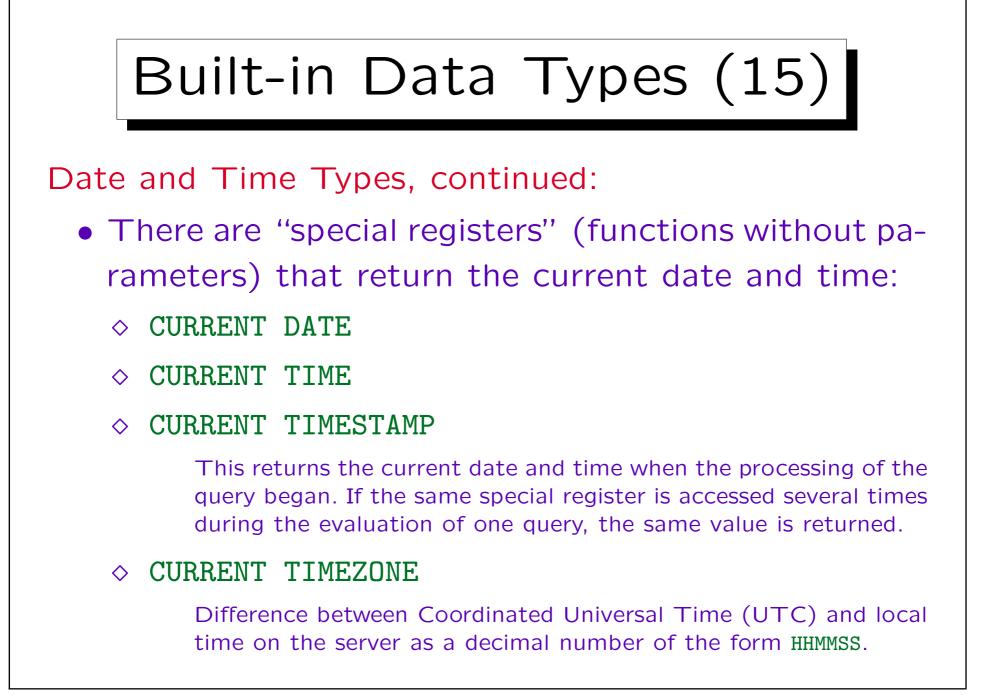
a table, these must be DMS tablespaces. The tablespace used for LOBs (LONG data) must be a large tablespace.

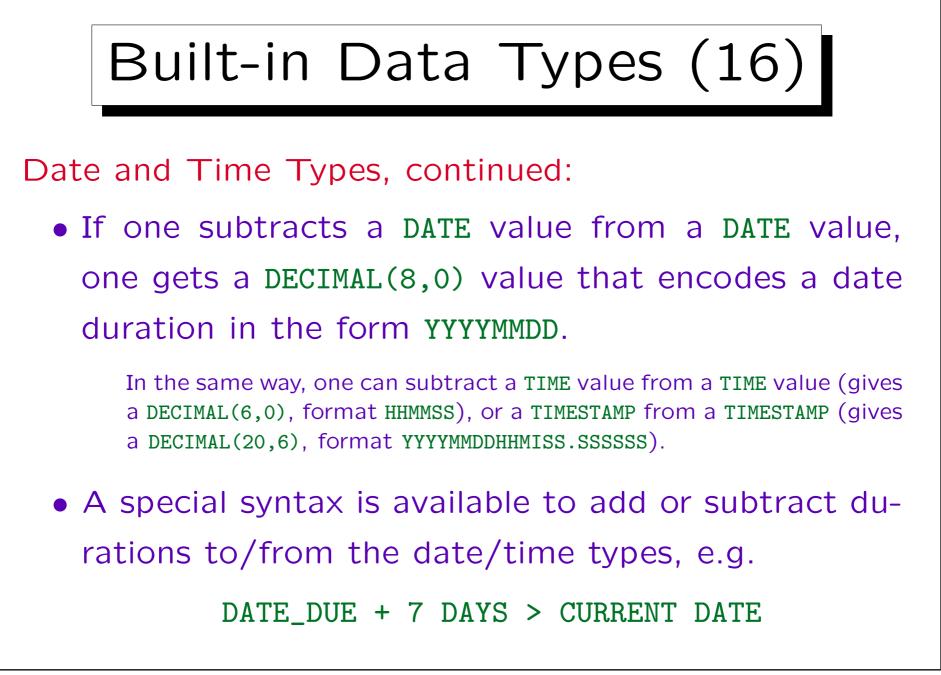












Distinct Types (1)

• Distinct types are one form of user-defined types (UDTs). They are defined based on a built-in type:

CREATE DISTINCT TYPE EX_NO AS INTEGER WITH COMPARISONS

The keywords "WITH COMPARISONS" are required for all types that support the standard comparison operators =, <>, <, >, <=, >= (i.e. all types except the LOB and LONG types). The keywords are supposed to remind the user that the comparison operators are inherited from the base type. Other operations like "+" are not automatically inherited.

• This means that EX_NO will internally have the same set of values as the built-in type INTEGER.

It is not possible to exclude e.g. negative numbers.



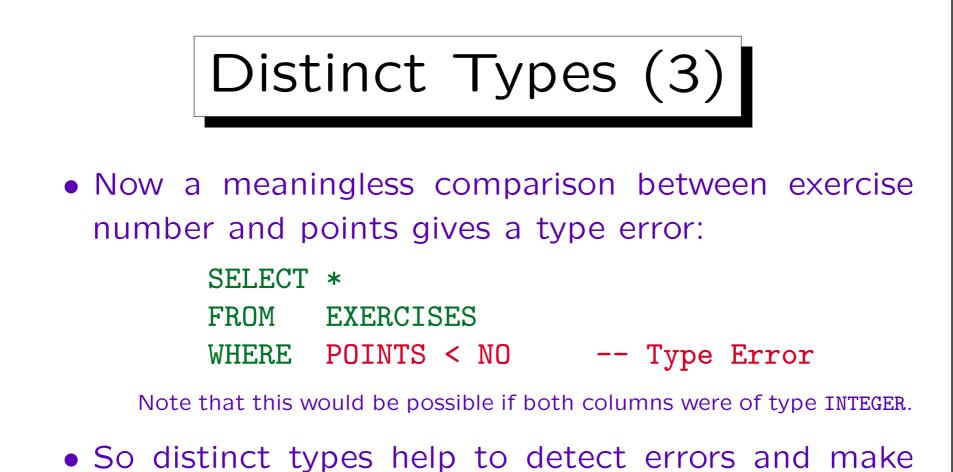
 The values of the new type EX_NO are considered as distinct from the values of the source type INTEGER, and the values of any other distinct type:
 CREATE DISTINCT TYPE POINTS AS INTEGER WITH COMPARISONS

It is possible to have a column and a distinct type with the same name. But it might be clearer to name types e.g. with the suffix " $_T$ ".

• Table columns can be declared with a distinct type:

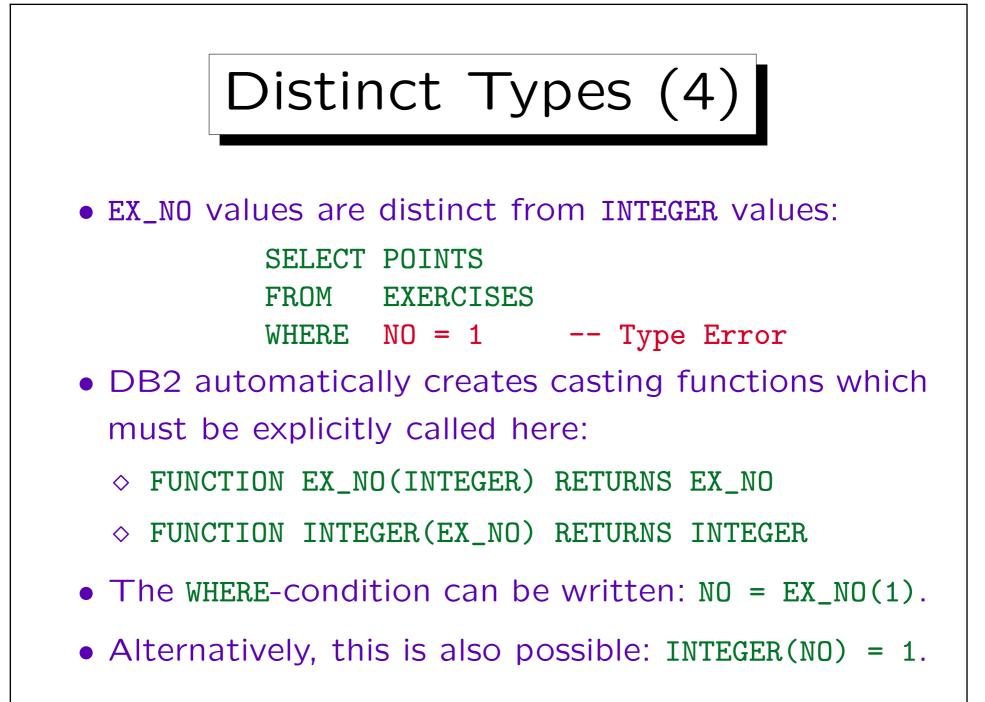
CREATE TABLE EXERCISES (

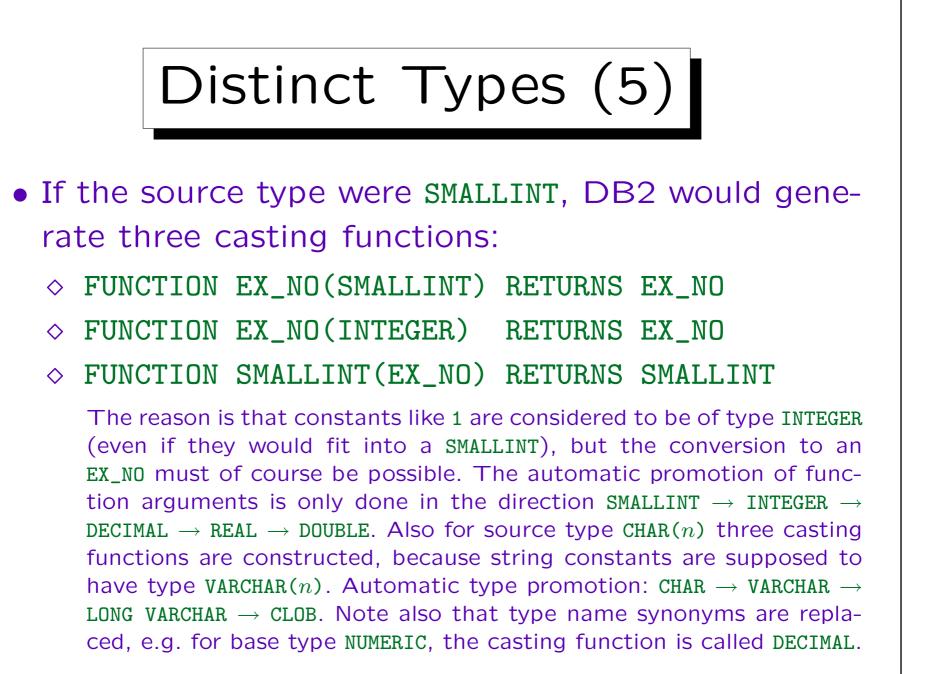
NOEX_NONOTNULLPRIMARYKEY,TOPICVARCHAR(40)NOTNULL,MAXPTPOINTSNOTNULL)

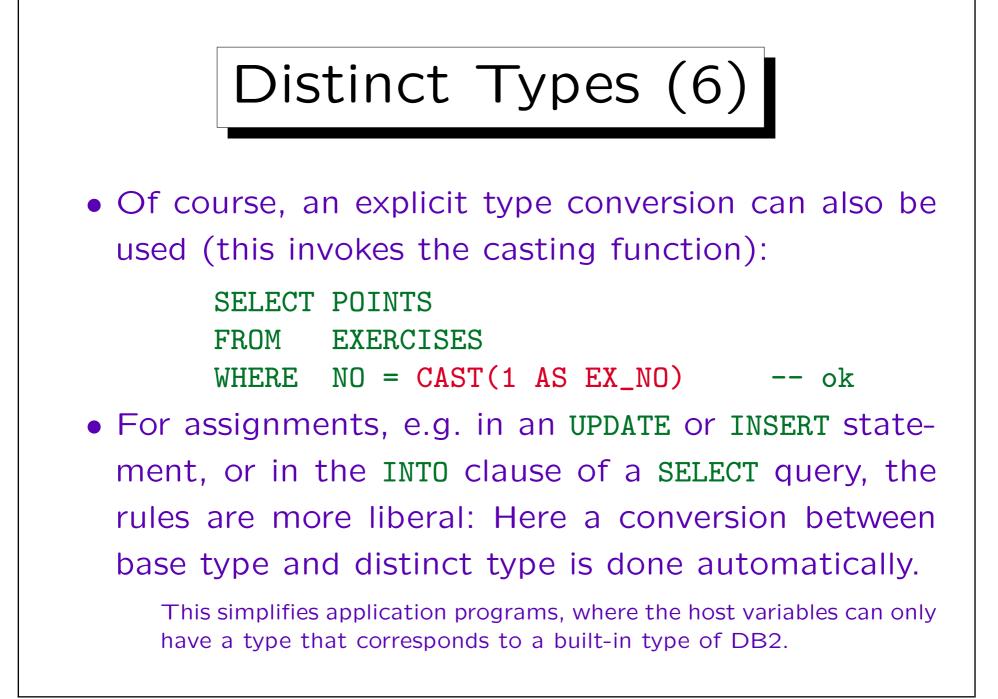


the semantics of columns clearer.

Furthermore, the source type is defined only in one place. If we later in the design think that a SMALLINT would suffice, this can be easily changed in the schema creation SQL script. However, once the tables are created and filled, this change is not possible.



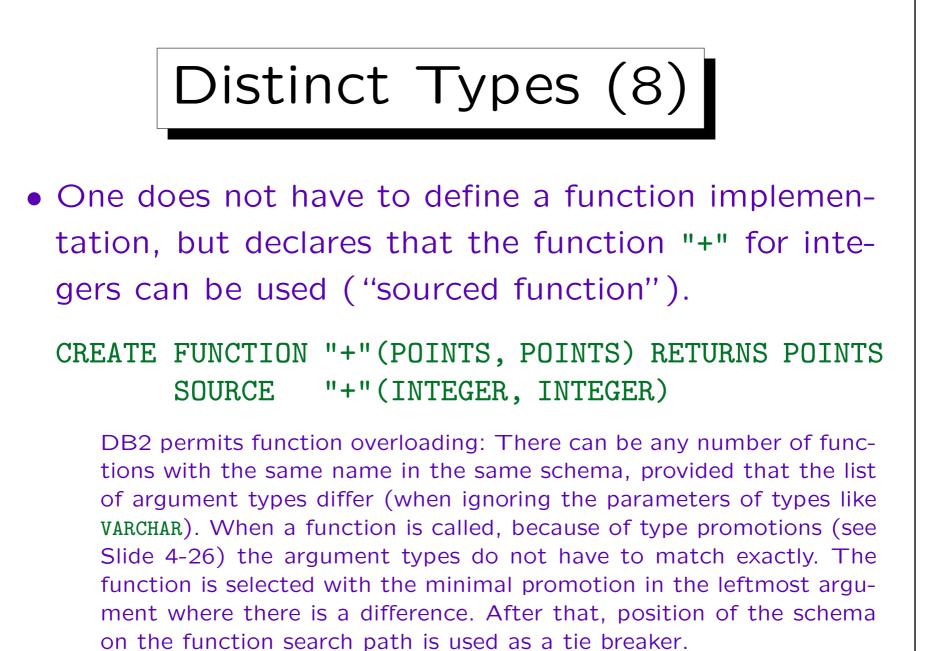


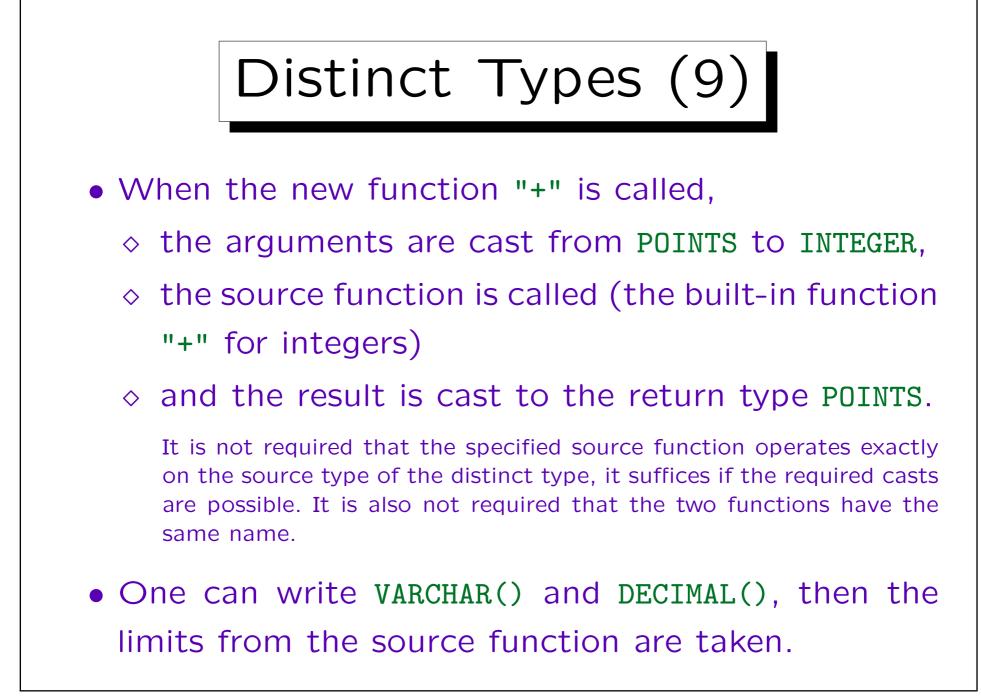


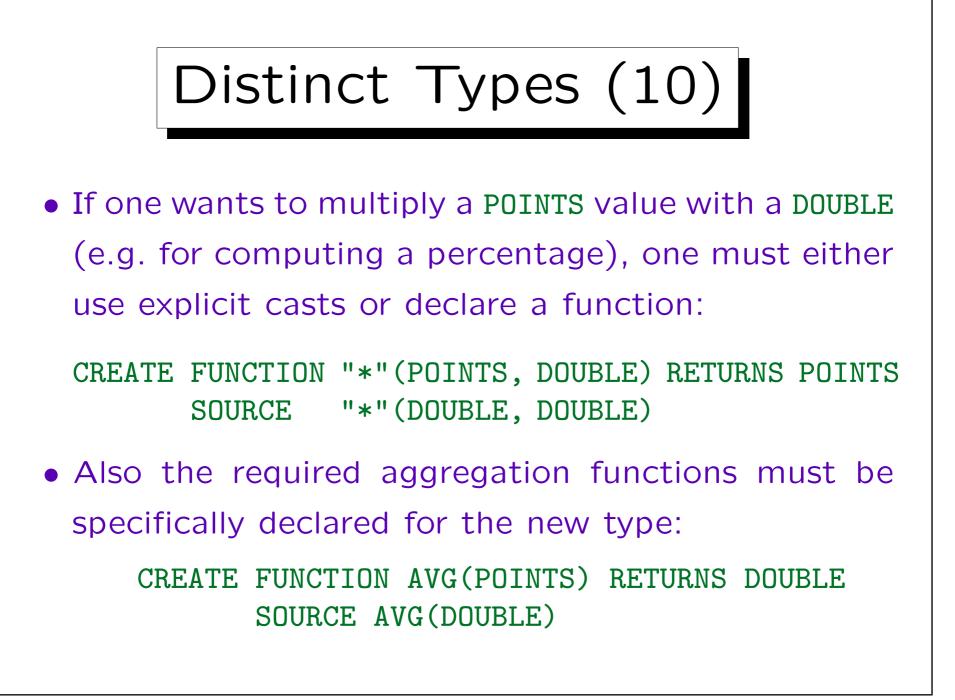


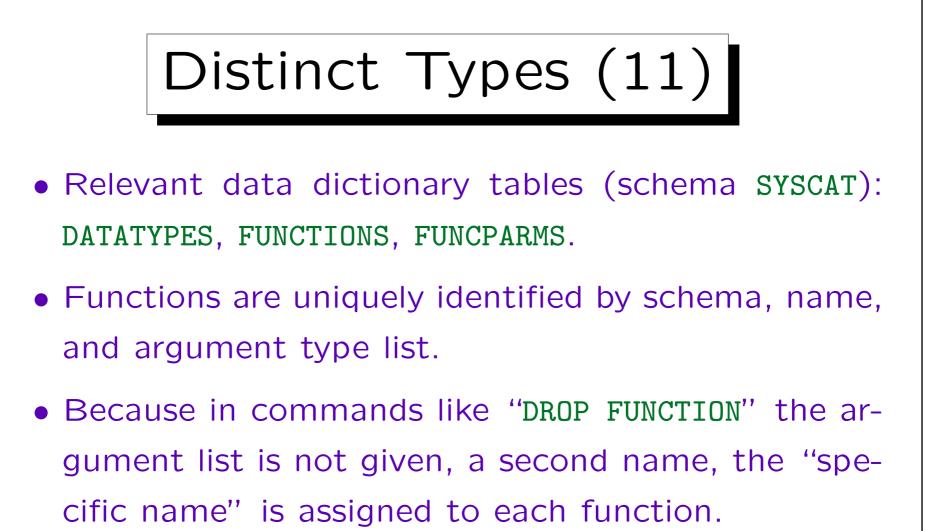
- No data type functions (except =, <, ...) are inherited from the source type to the distinct type.
- E.g. one cannot add exercise numbers (which is good, because this would not make sense).
- However, it would make sense to add points.
- Then it is necessary to explicitly declare a userdefined function that performs addition of points.

This function can be named "+", so that the infix operator + can also be used for points. DB2 supports the function call syntax also for operators when these are put into "...", e.g. one can write SELECT "+"(1,2) FROM

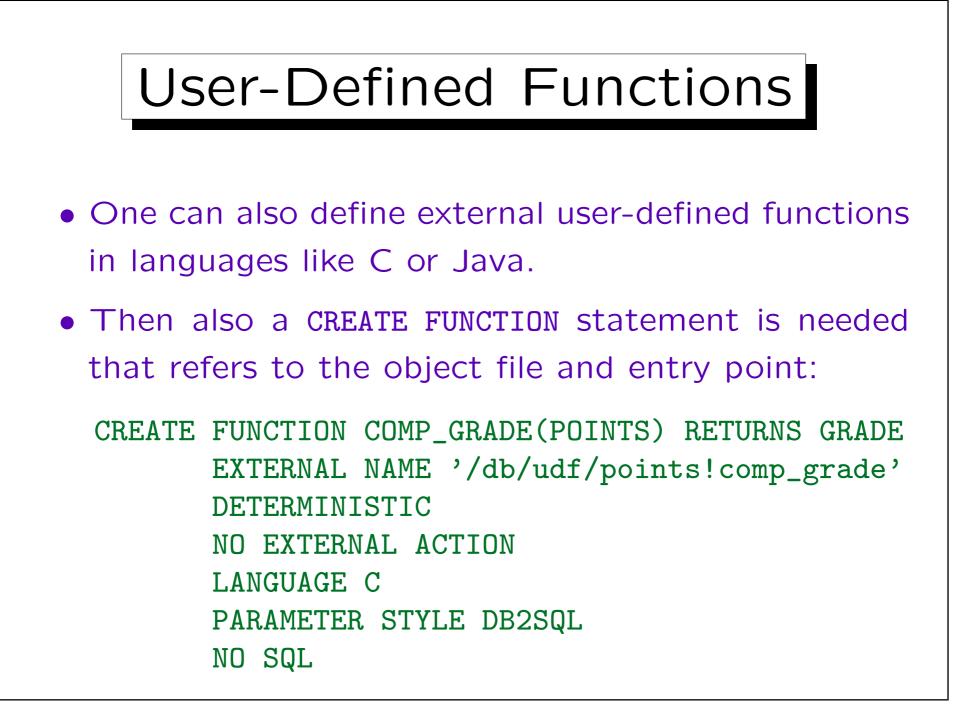








Functions cannot be called with the specific name. One can define a specific name after the return type with the keyword SPECIFIC. Otherwise the DBMS chooses one (see data dictionary).





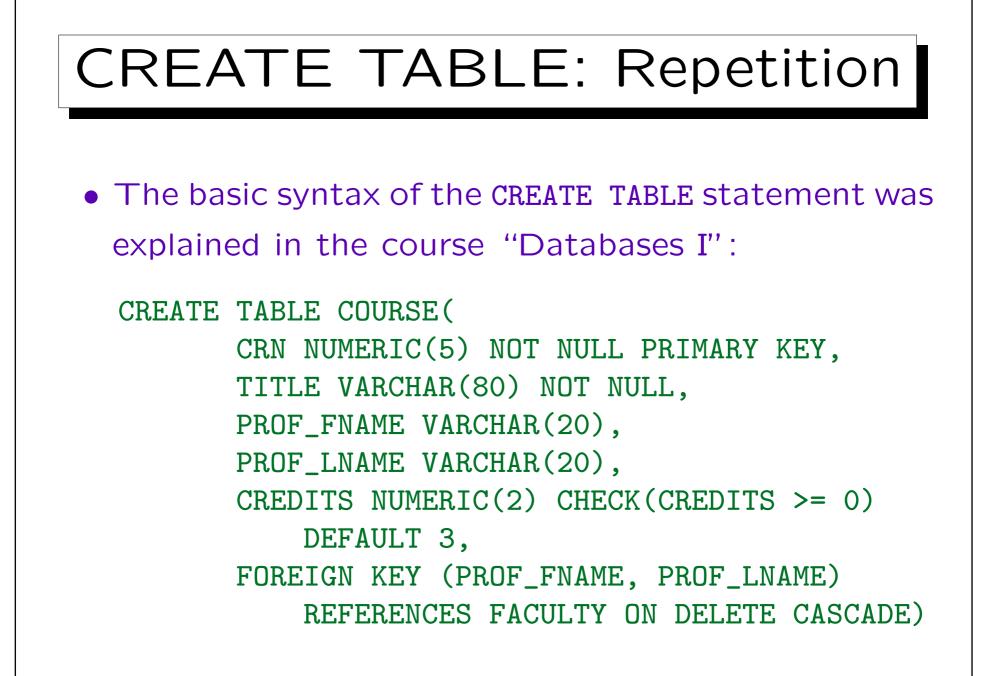
1. Data Types

2. Tables

3. Indexes

4. Triggers

5. Typed Tables





- In DB2, one can write WITH DEFAULT instead of only DEFAULT (as in the standard), but the keyword WITH is optional.
- In DB2, it is possible not to write a value after the keyword DEFAULT. Then the number 0, the empty string etc. is chosen ("system default value").

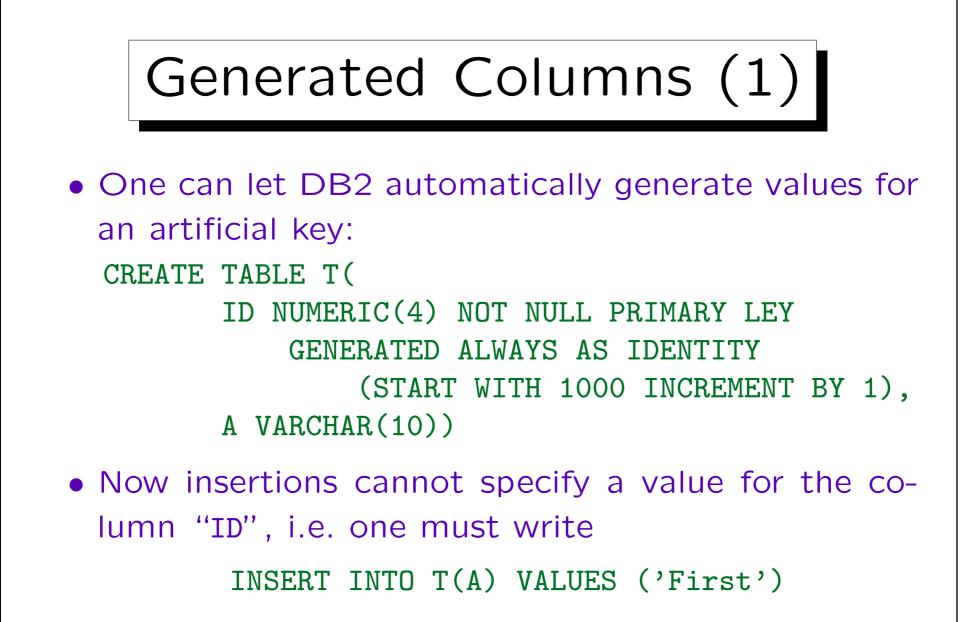
For CHAR(n), the empty string means of course n blanks. For date/time columns, the earlierst possible value is taken.

• However, if one does not use the DEFAULT specification at all, the default value is NULL.

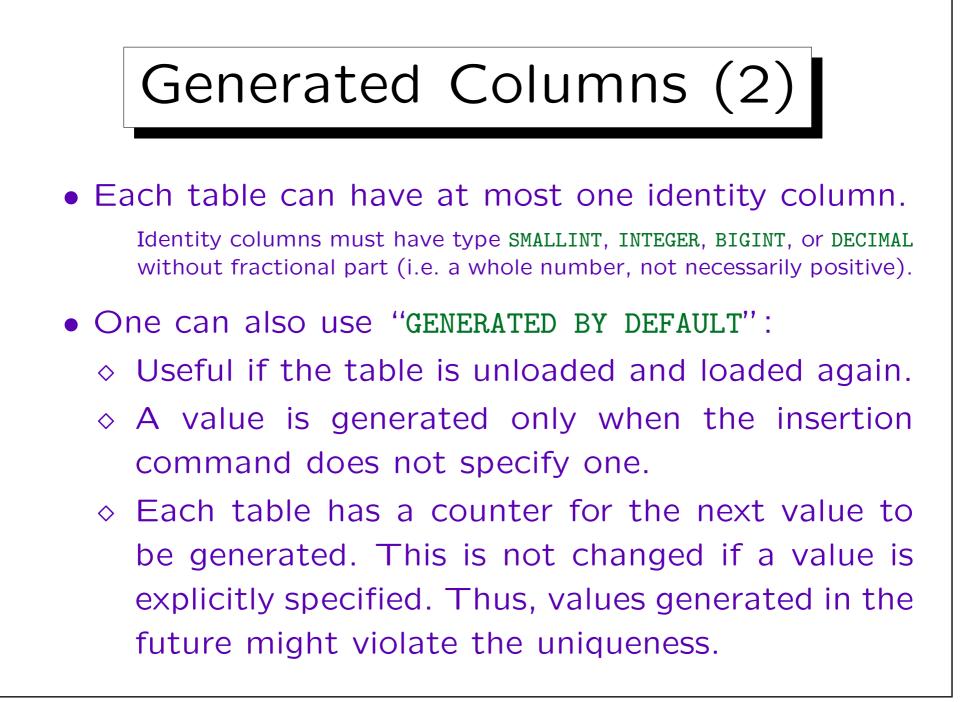
Default Values (2)

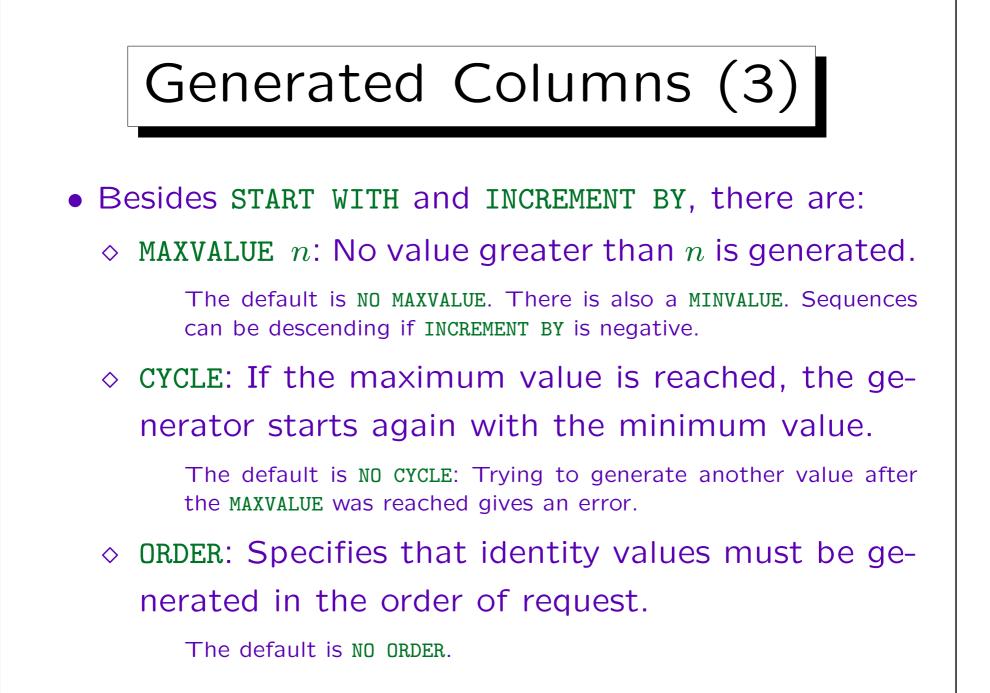
- Note that specifying a non-null default value does not prevent the insertion of a null value, one only has to explicitly specify NULL in the VALUES list.
- The only safe way to prevent null values is to use the NOT NULL column constraint.

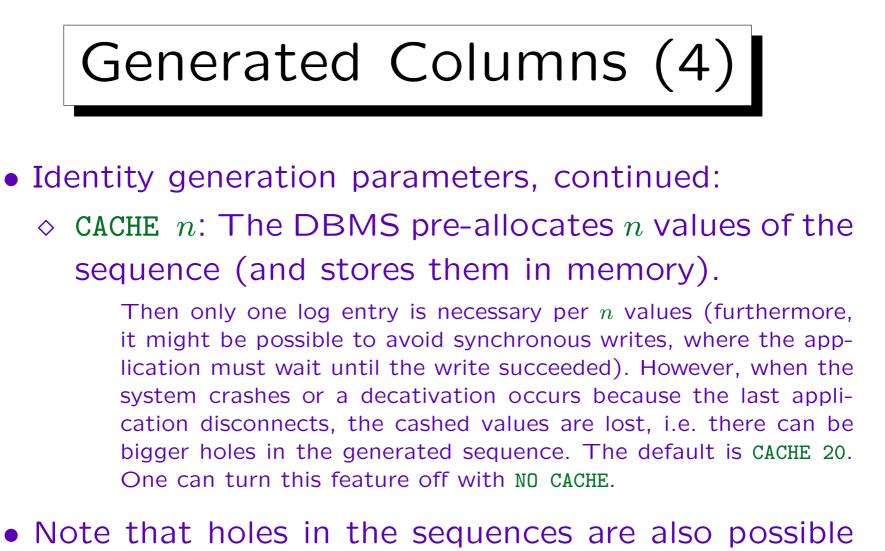
One could also use a CHECK-constraint with an "IS NOT NULL" condition, but then DB2 still does not allow to use the column in a primary or unique key (which requires that the column is not nullable).



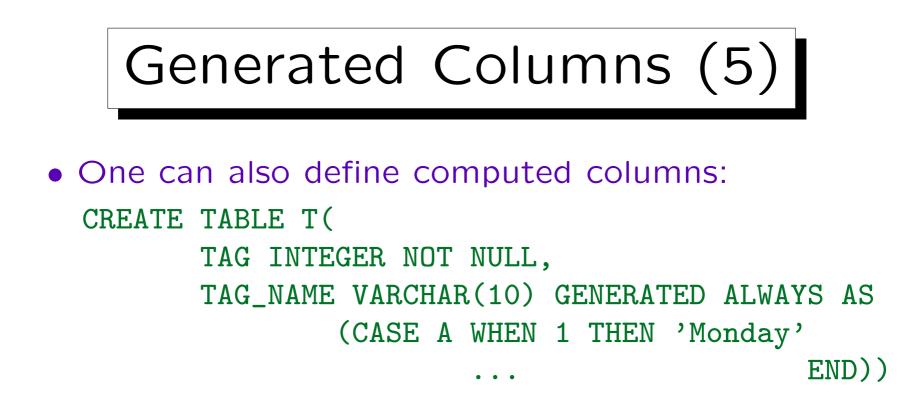
• This will insert the row (1000, 'First').





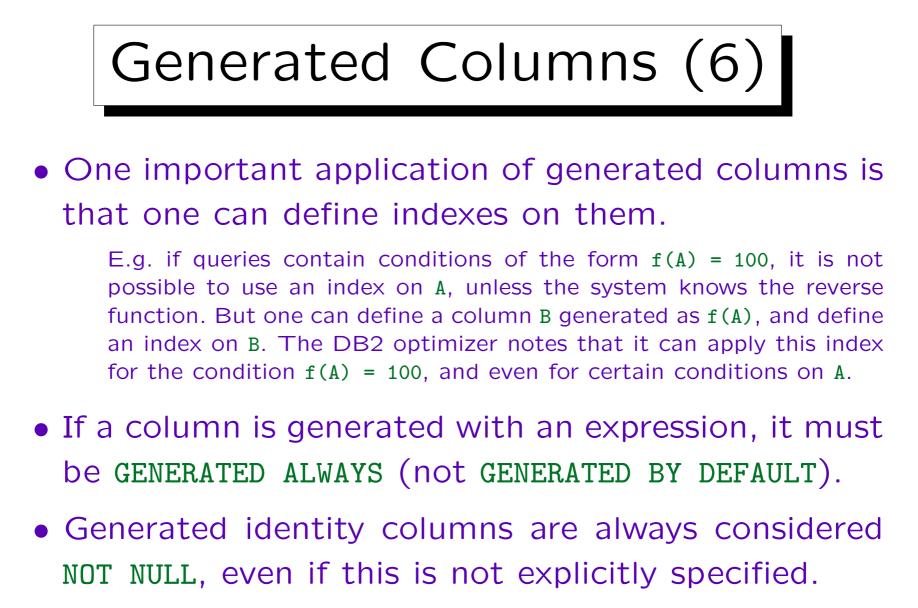


• Note that holes in the sequences are also possible when a transaction is rolled back: Once a value was generated, it is not recycled.

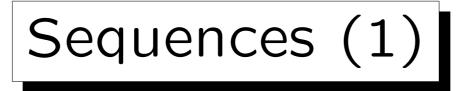


• However, it seems that a view would be preferable in this example.

It is possible to call user-defined functions to compute the generated value (under certain conditions, e.g. without external action). If that function needs a lot of runtime, the generated column might be the better alternative. The defining expression cannot use subqueries or special registers (e.g. CURRENT DATE is not allowed).



And even if it is only generated BY DEFAULT.



- A sequence works like the generator for identity columns, but it is not tied to a specific table.
- Thus, the generated values can be used in a more flexible way.
- When a sequence is created, the same parameters as for identity columns can be specified:

CREATE SEQUENCE S AS DECIMAL(5) START WITH 100 INCREMENT BY 1

• If AS $\langle Type \rangle$ is left out, INTEGER is assumed.



• One gets the next value for a sequence with the expression:

NEXT VALUE FOR S

(where S is the name of the sequence).

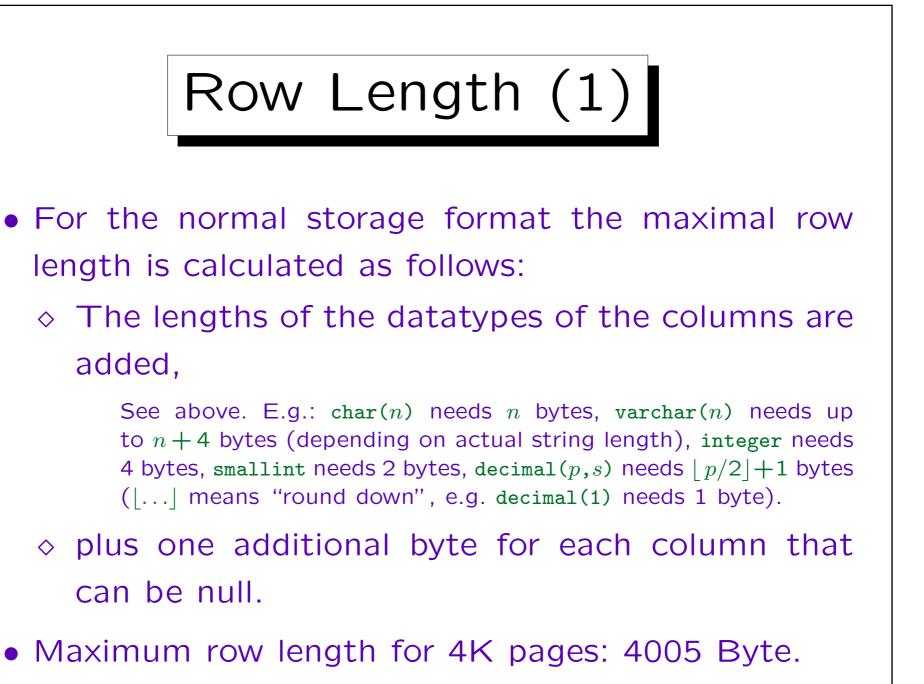
• The last generated value is returned by

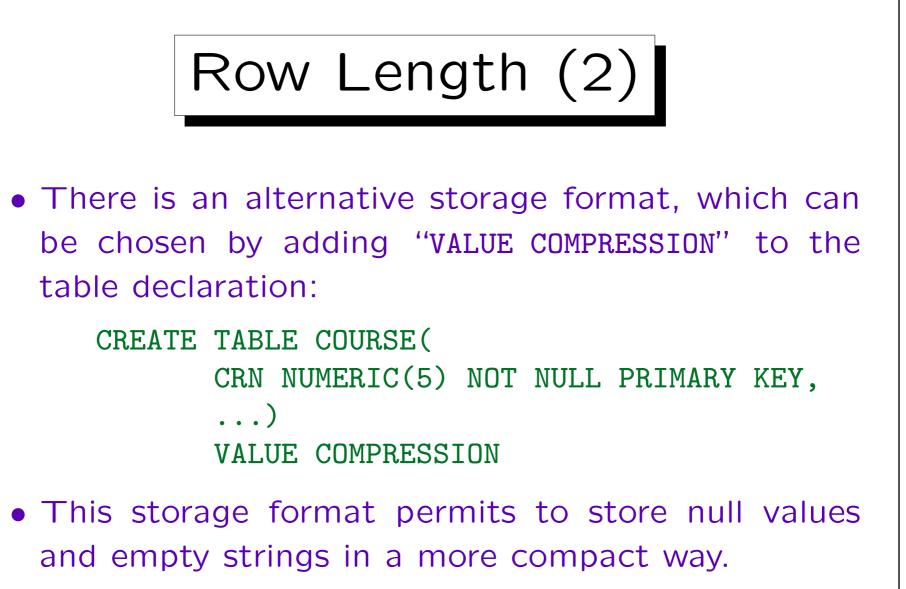
PREVIOUS VALUE FOR S

This is helpful if the same value must be entered in two or more places (e.g. an invoice numer is needed also in all line items).

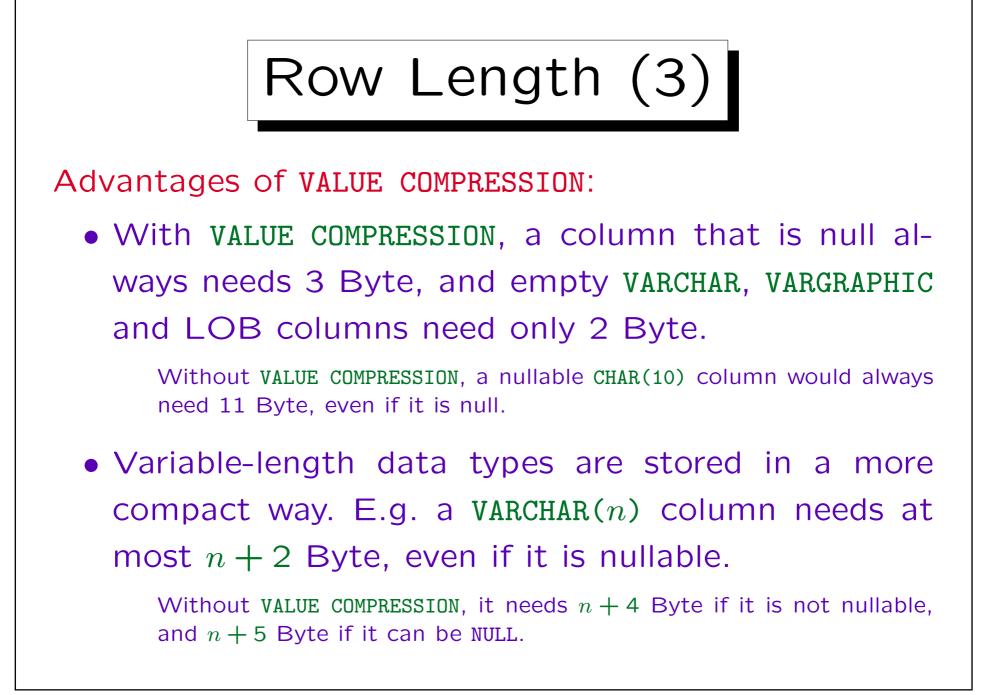
• A sequence can be altered, e.g. with

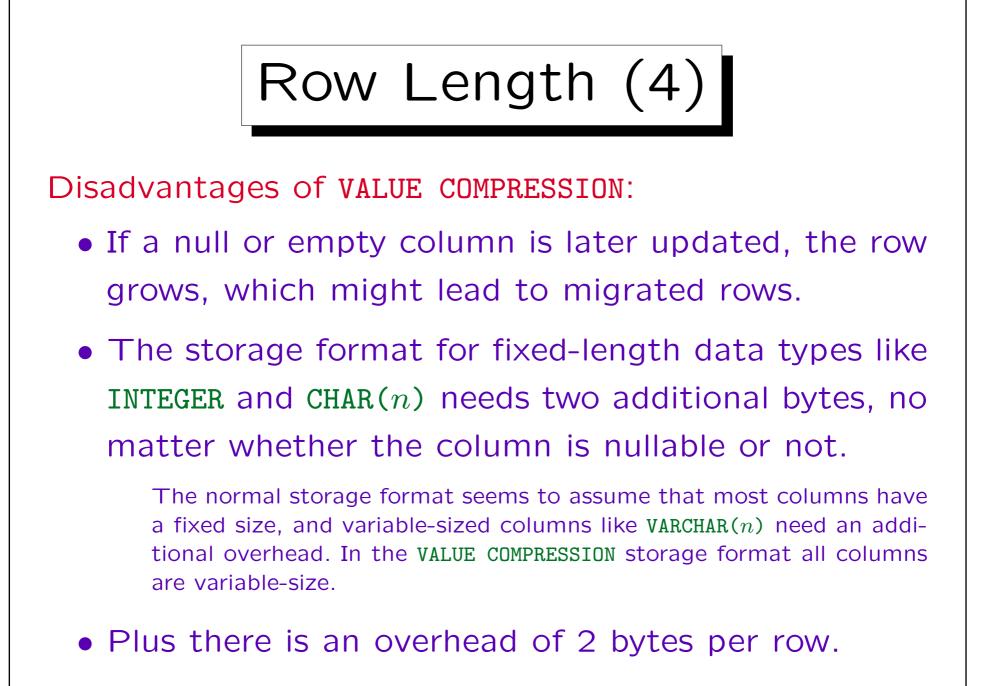
ALTER SEQUENCE S RESTART WITH 200

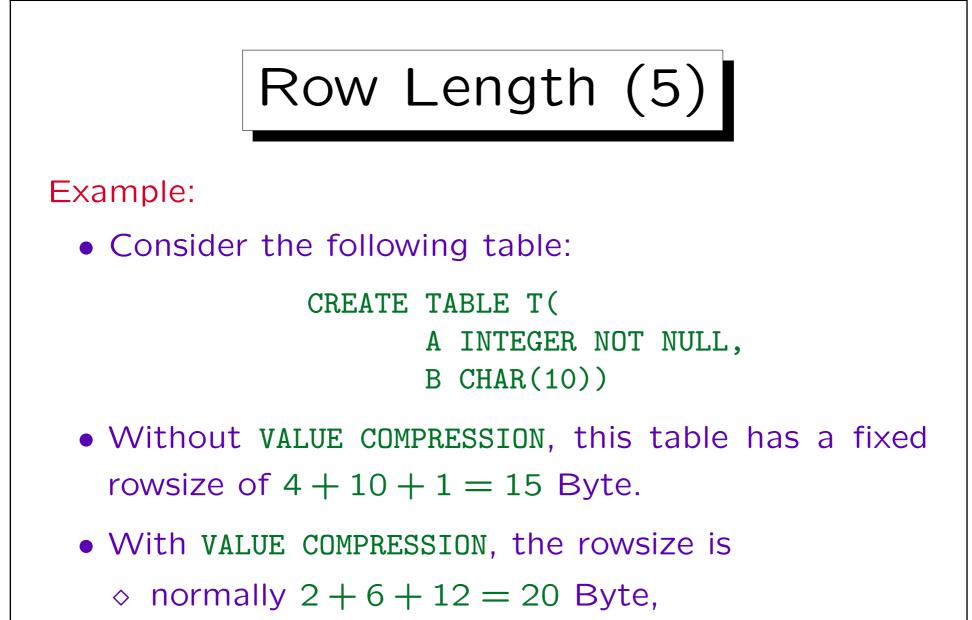




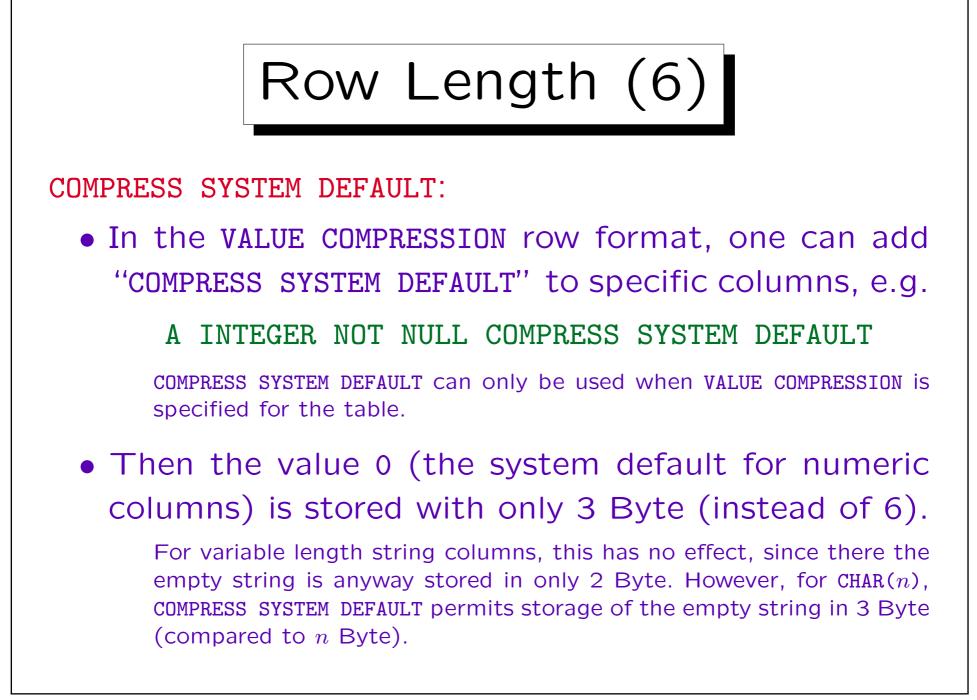
It is not real value compression, only a different stoarge format for rows. In the meantime, DB2 also has a real compression.

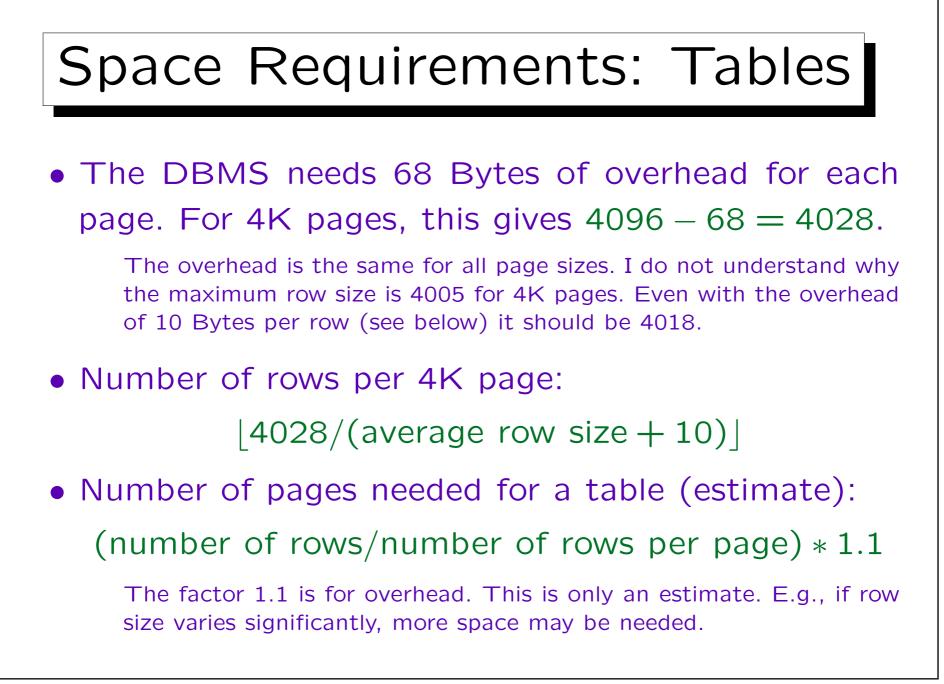


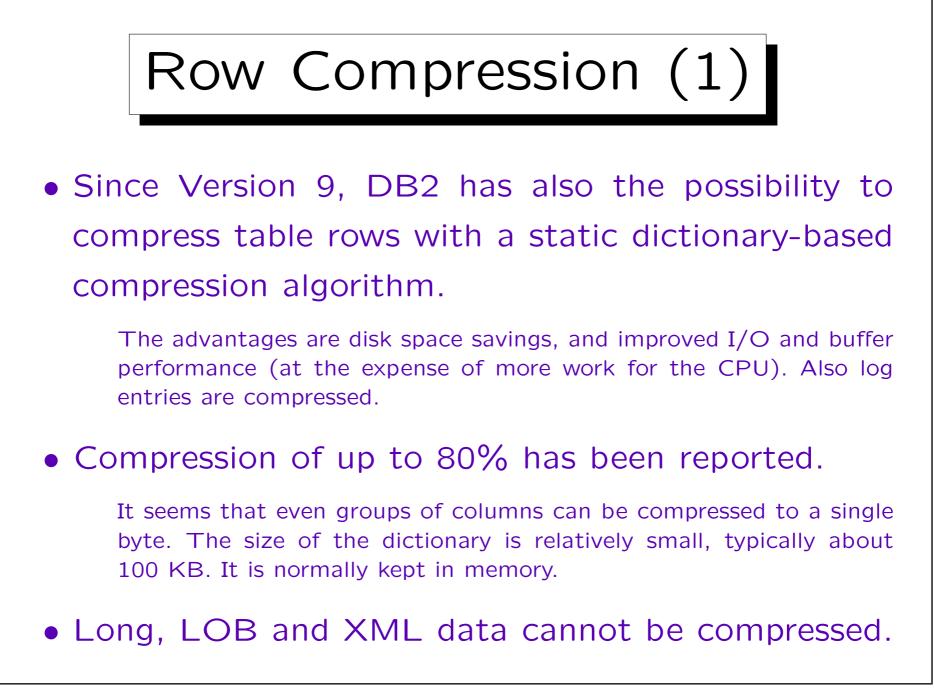


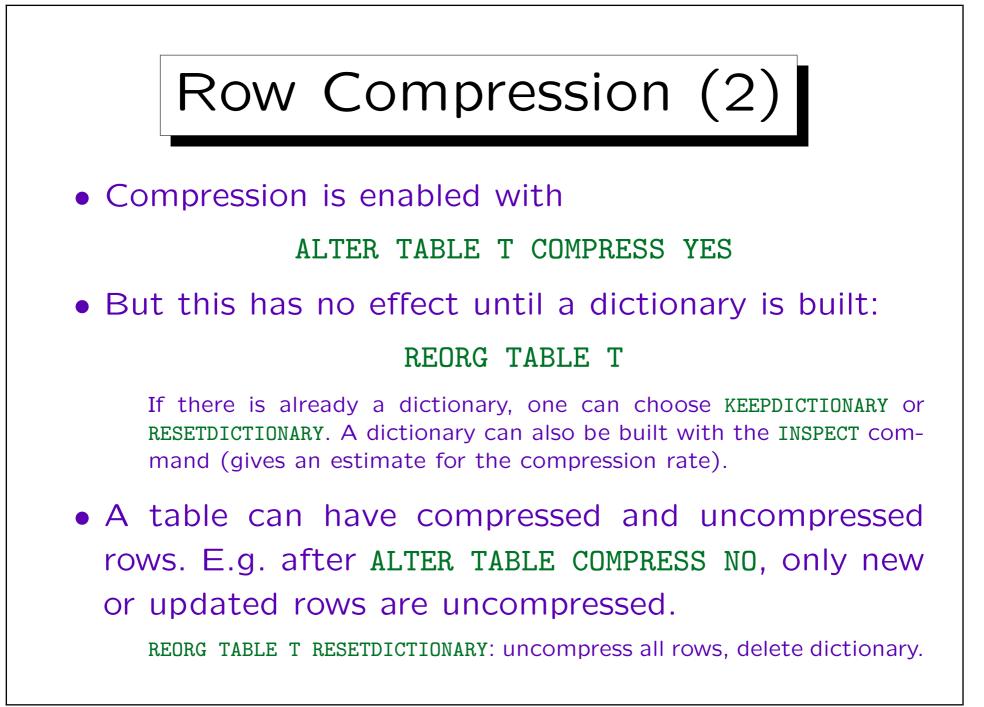


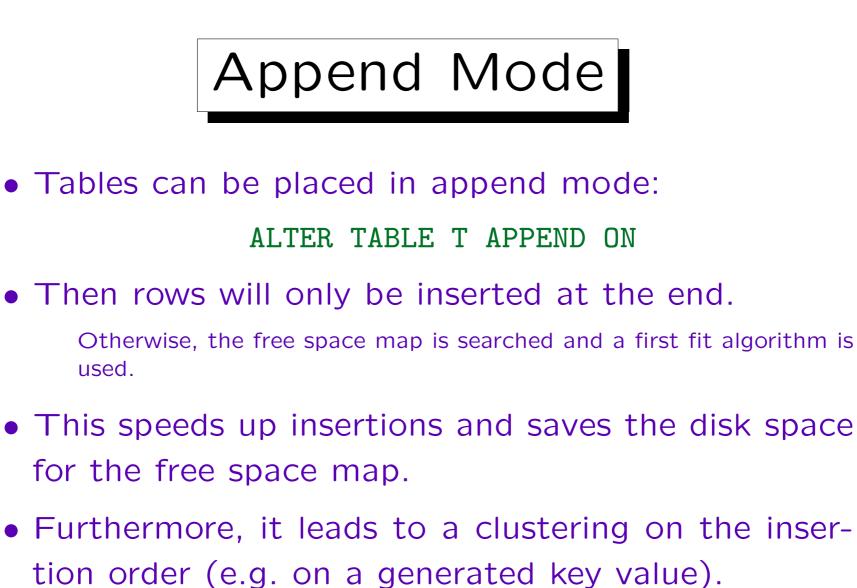
♦ but only 2 + 6 + 3 = 11 Byte if B is null.



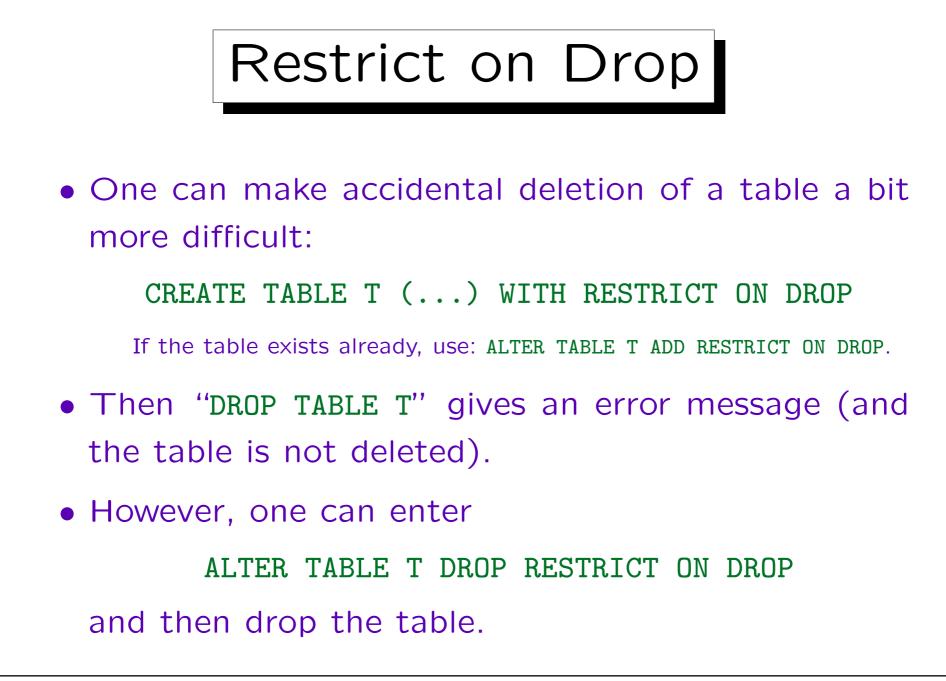


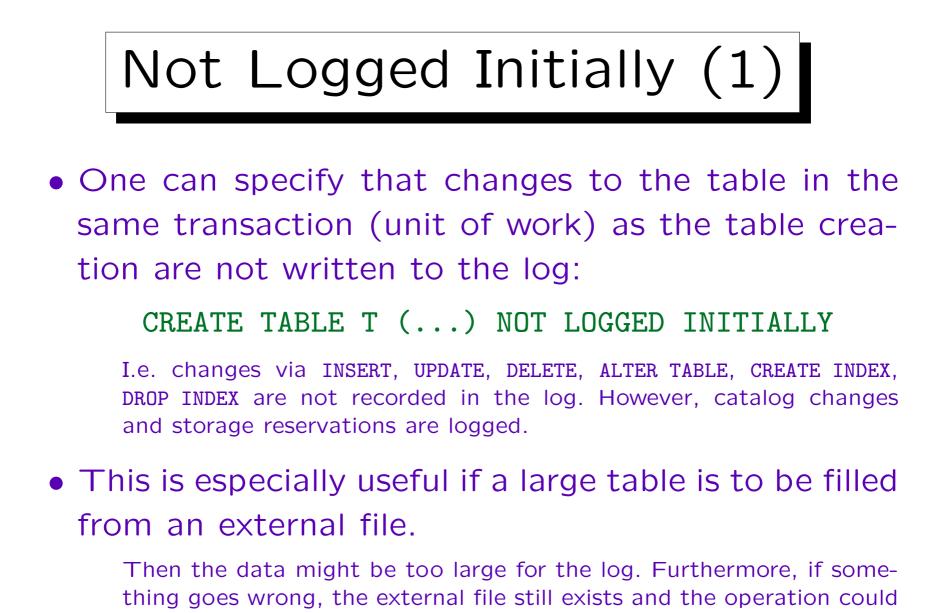


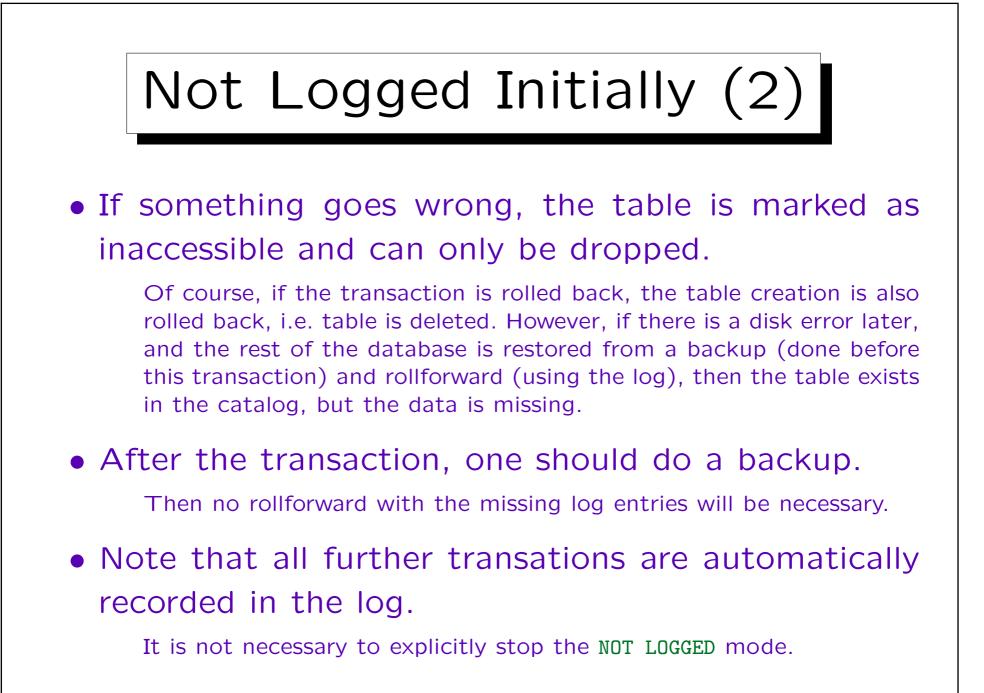


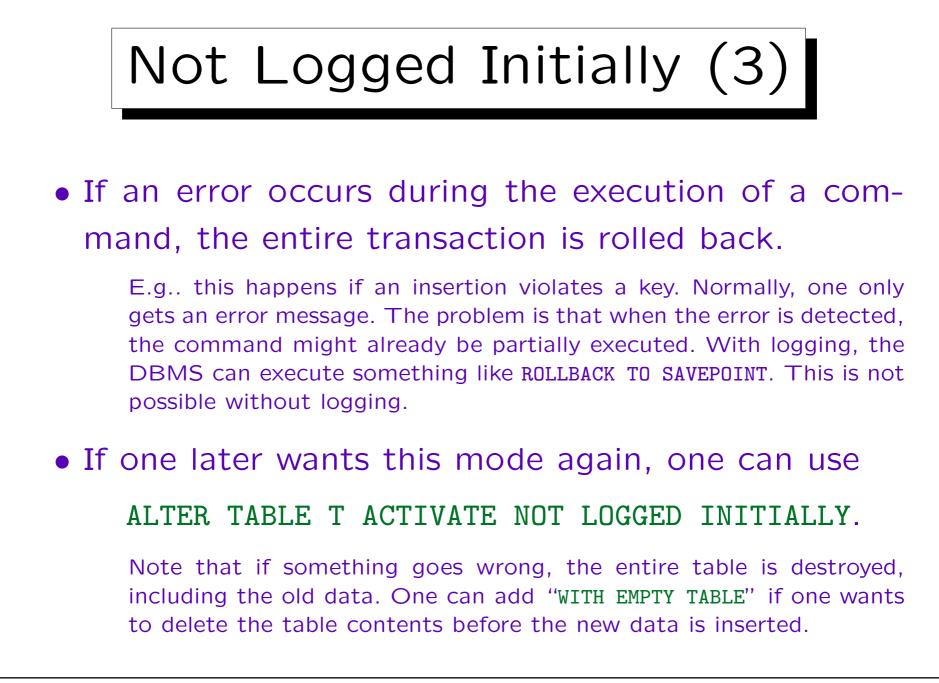


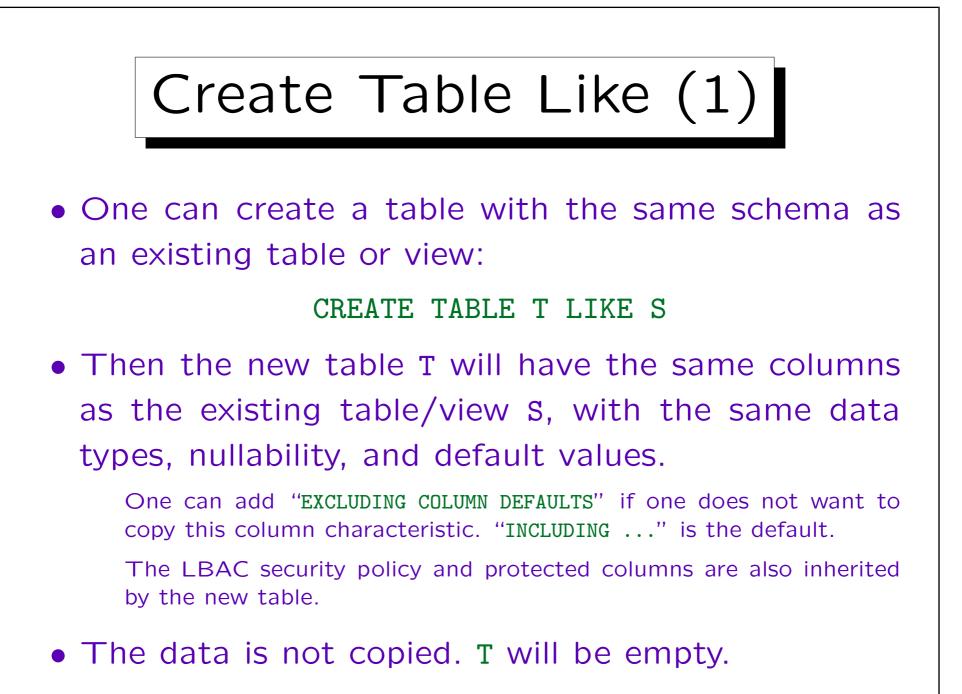
• Of course, if rows are deleted, holes remain.

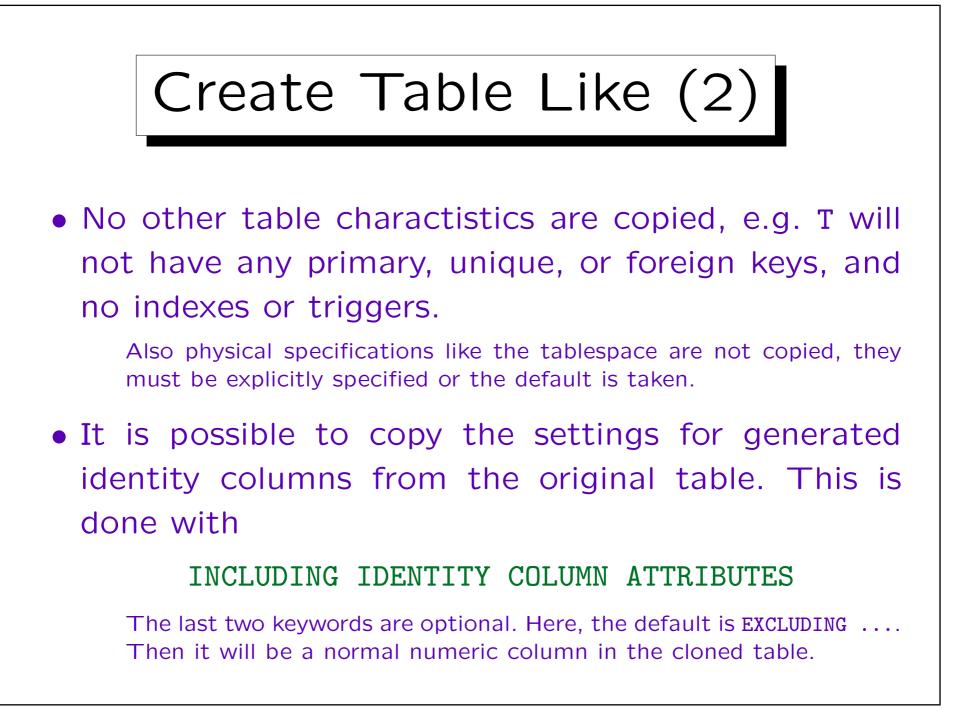


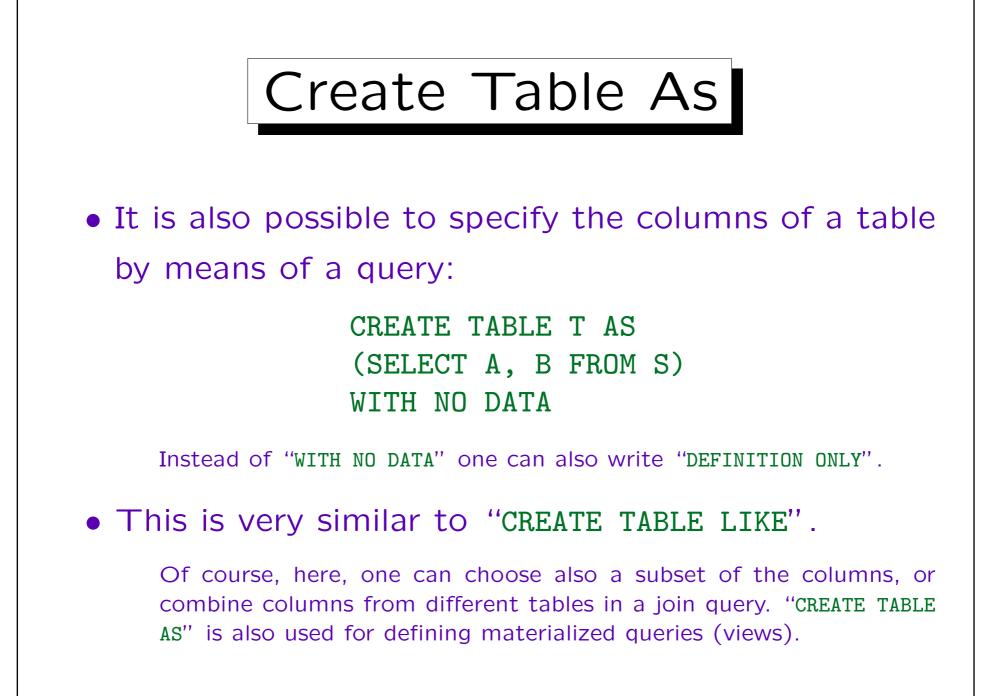


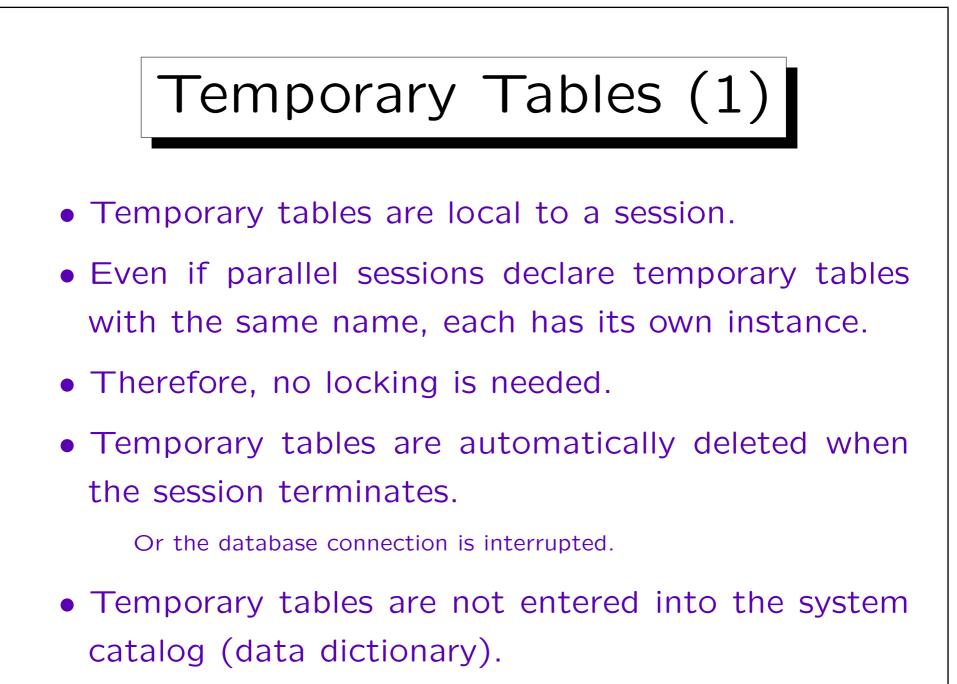


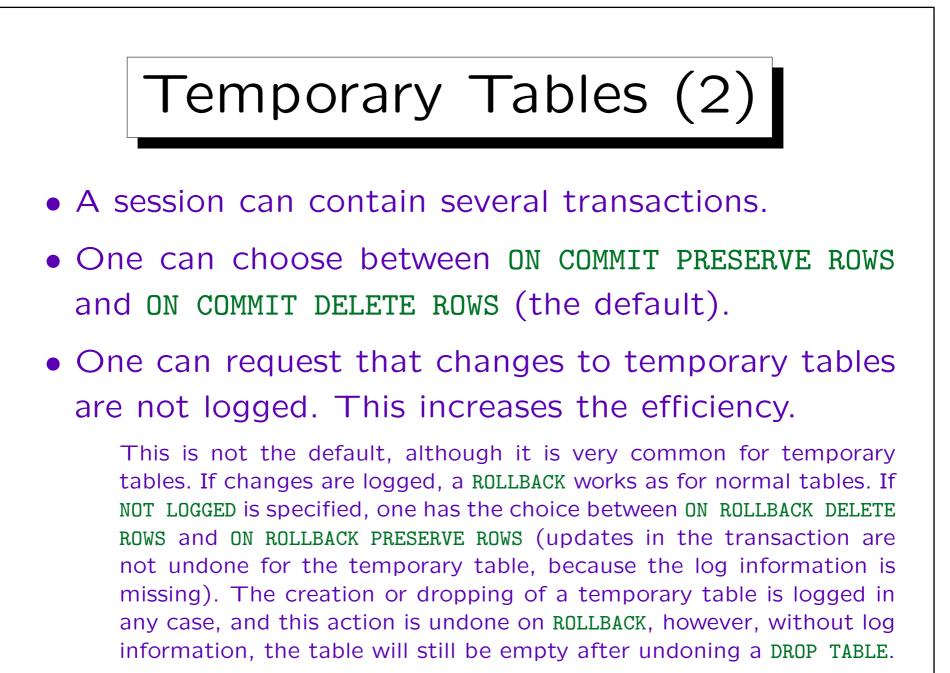


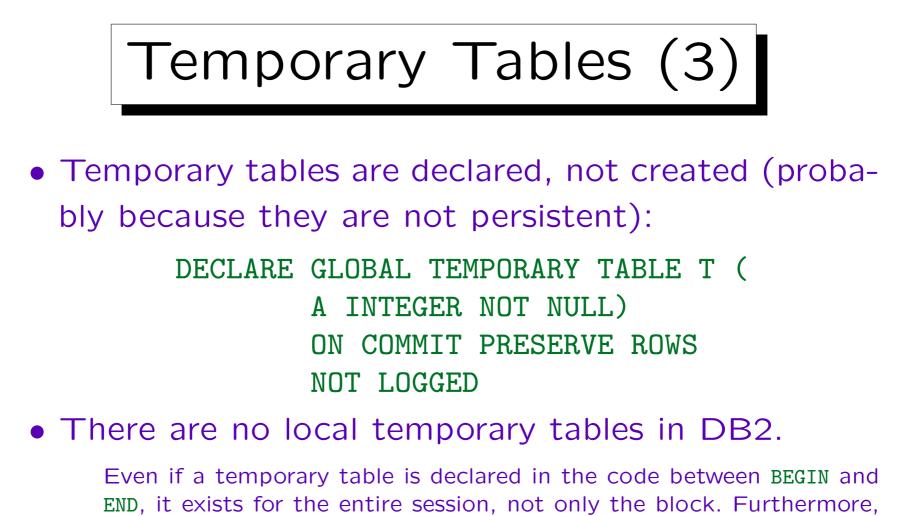






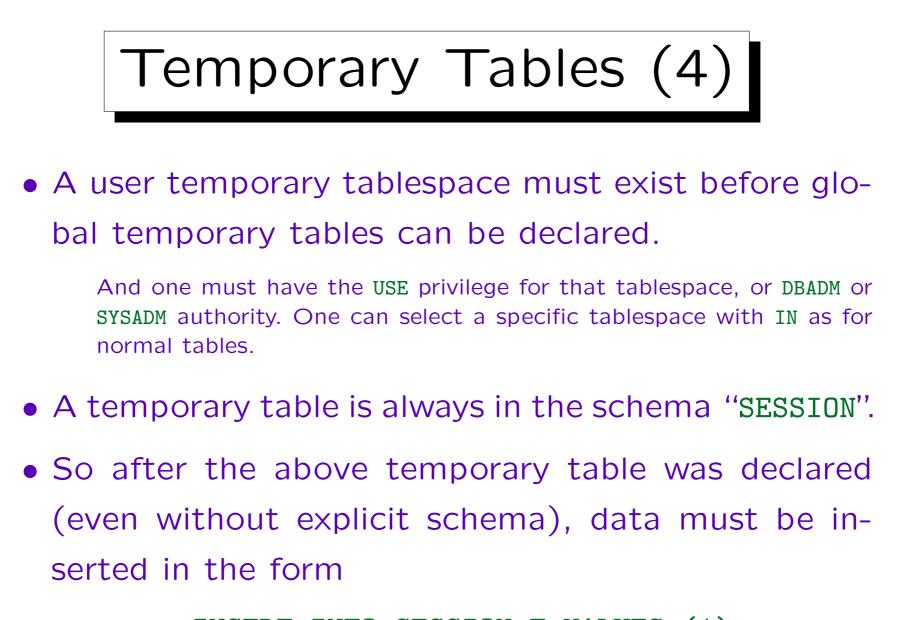






Even if a temporary table is declared in the code between BEGIN and END, it exists for the entire session, not only the block. Furthermore, the system sometimes uses internally temporary tables for the evaluation of a query (exist only locally during evaluation of a single query).

Another explanation might be that this syntax conforms to the SQL standard, and the standard has also other forms of temporary tables.



INSERT INTO SESSION.T VALUES (1)

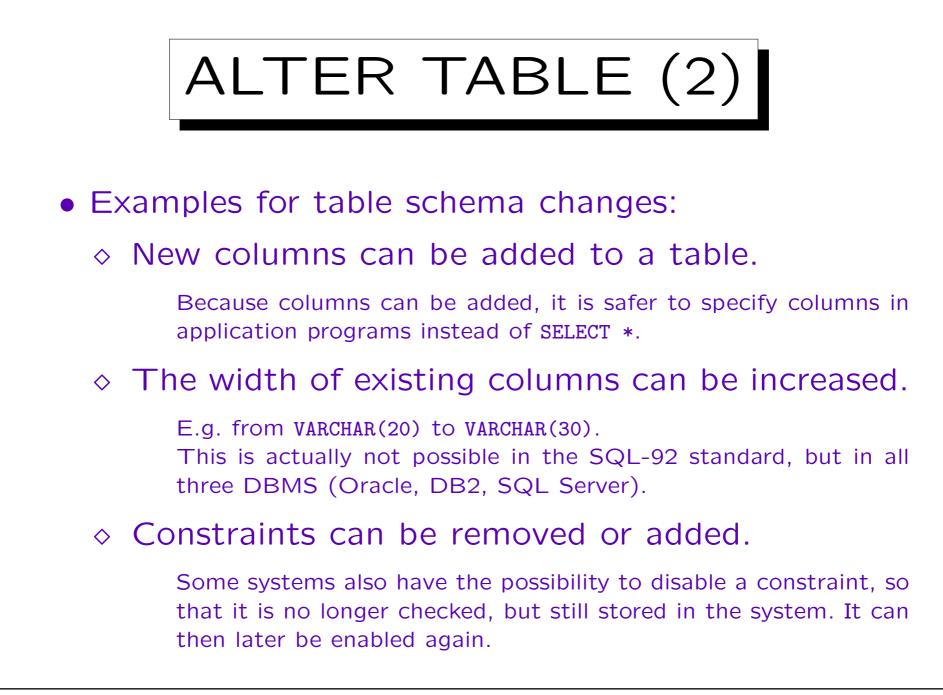
Temporary Tables (5)

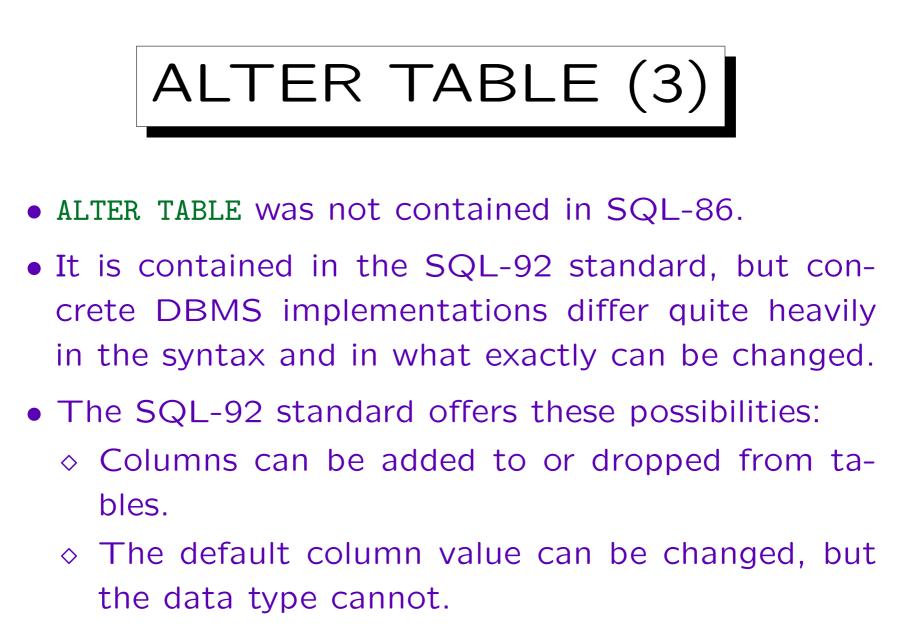
- If the clause "WITH REPLACE" is added to the table declaration, a temporary table with the same name is automatically dropped, if one should exist.
- Keys and foreign keys cannot be used for temporary tables.
- Columns of temporary tables cannot be of long, LOB, or XML type.
- One can declare a temporary table LIKE another table, or AS the result schema of a query.



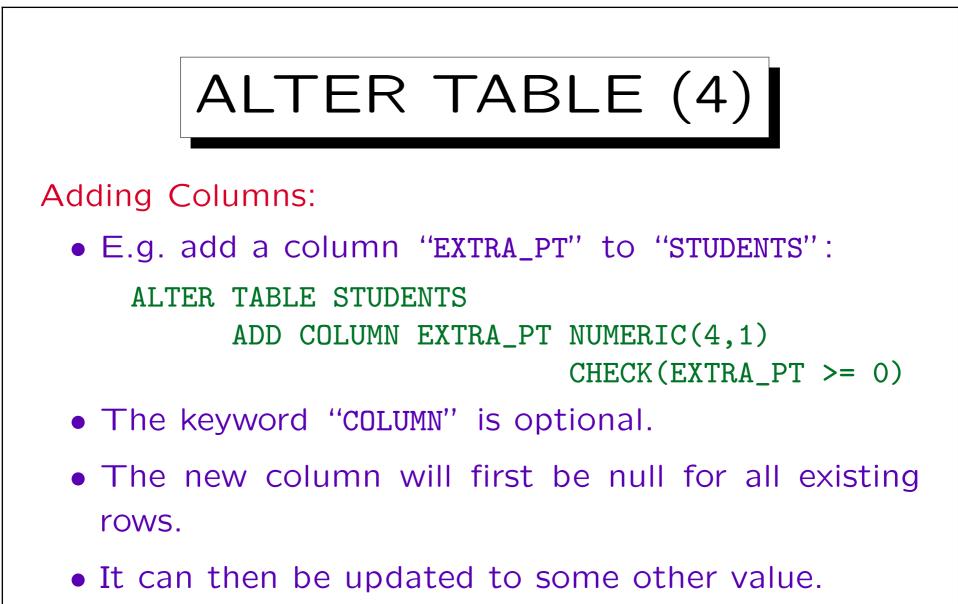
- The purpose of the ALTER TABLE command is to modify the schema of an existing table.
- In principle, one could copy the data to a temporary location, drop the table, create the table again with the modified schema, and copy the data back. But:
 - ◊ Copying the data is impractical for large tables.
 - Table entries may be referenced in foreign keys:
 One might end up recreating the entire DB.

Also indexes, grants, views, triggers, stored procedures etc. reference tables. Some of this information will be lost when the table is dropped.

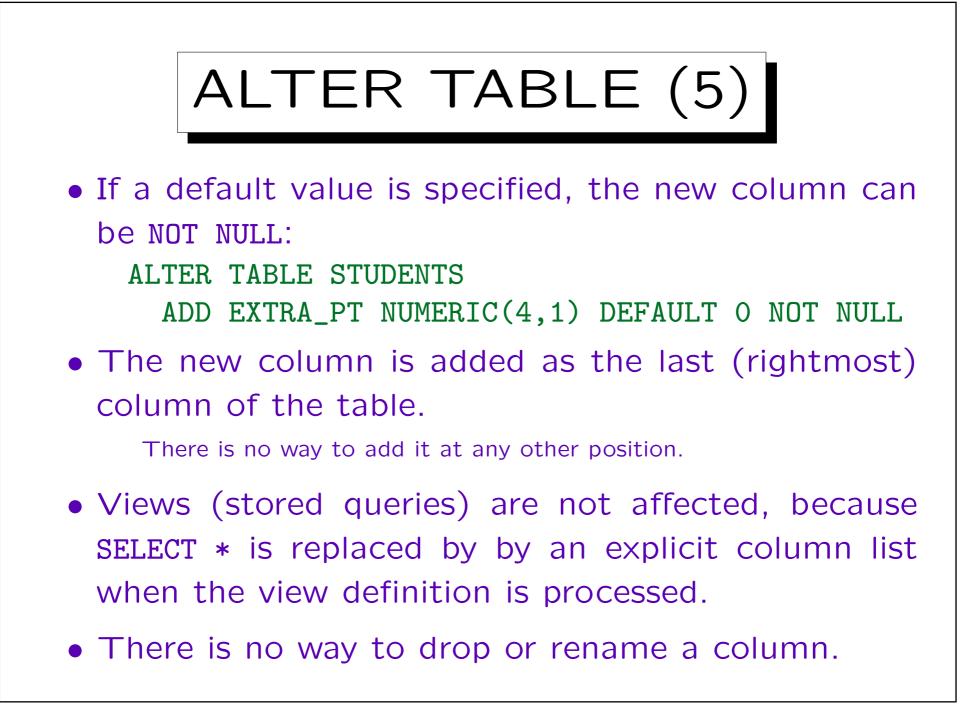


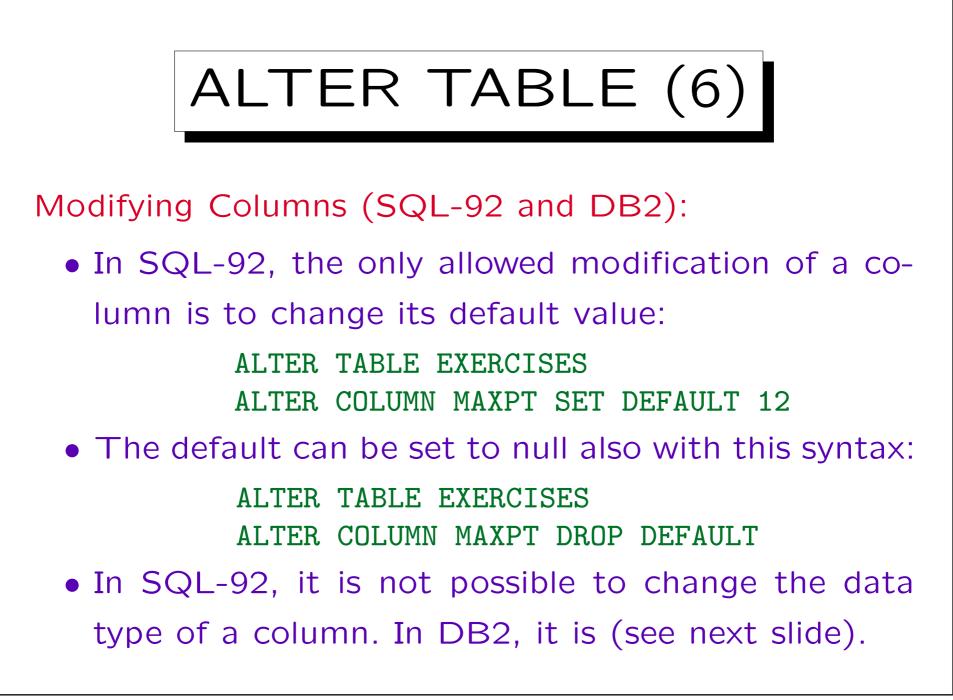


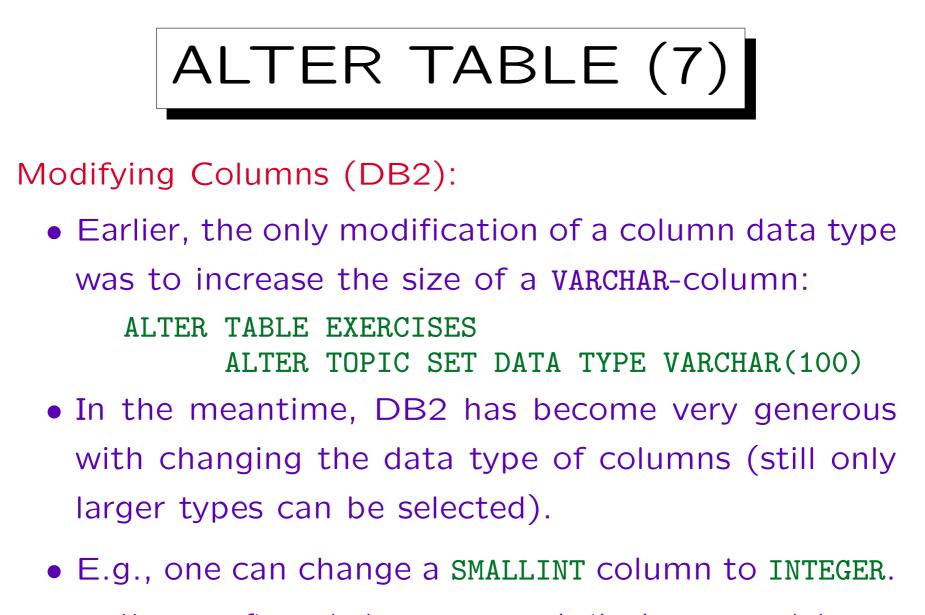
◊ Constraints can be added or removed.



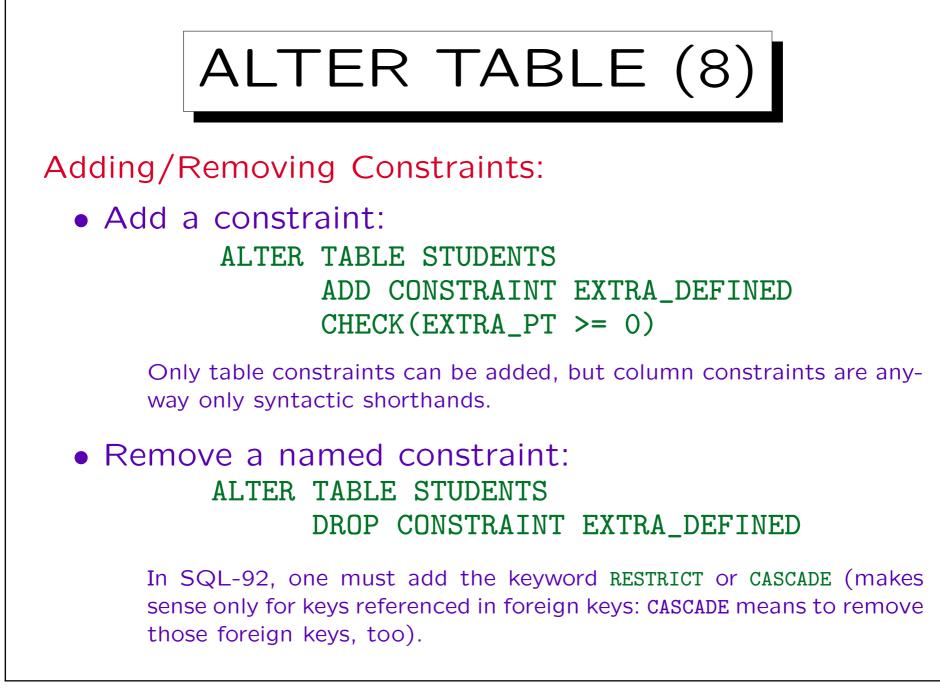
It might, however, create efficiency problems if the rows become much longer than they were when the were inserted.

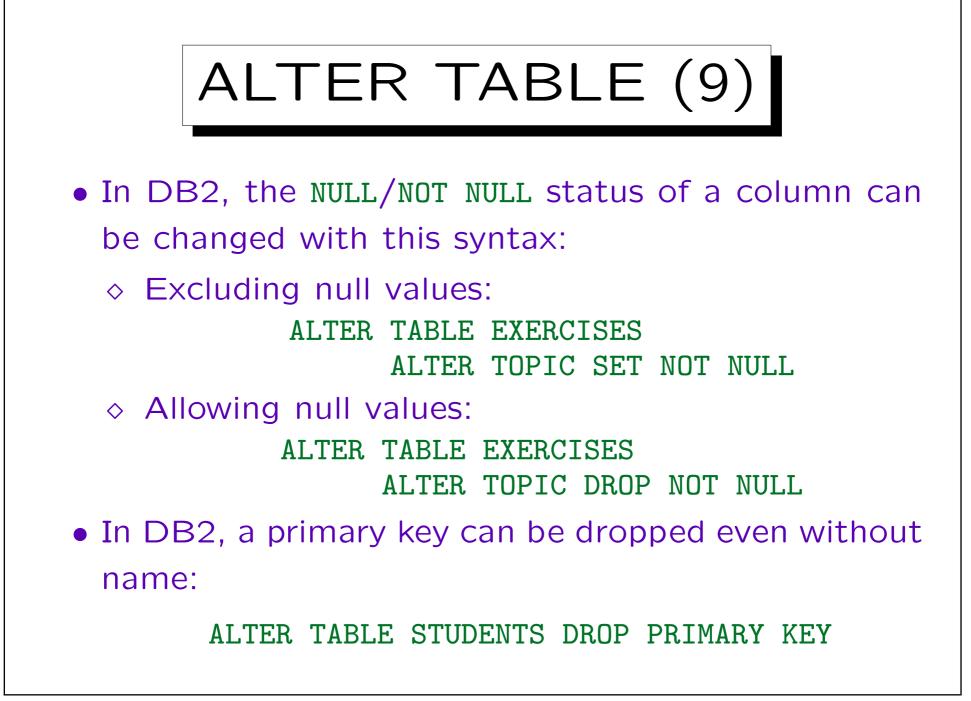






However, after such changes a reorganization is recommended.









2. Tables

3. Indexes

4. Triggers

5. Typed Tables

Stefan Brass: DBA-Zertifizierung

Indexes (1)

- Indexes permit to access rows with a given value for a column without scanning all rows in the table.
- Indexes are created on a specific column or column combination of a specific table:

CREATE INDEX X ON T(A)

• A typical data structure for indexes are B^+ trees.

The leaf nodes contain all values of the column in sorted order together with pointers to the corresponding rows (RIDs). RIDs are also called ROWIDs or TIDs (tuple identifier).

• Indexes and their use in query evaluation have been discussed in the course "Databases IIB".

Indexes (2)

 As usual, indexes can be declared as unique, meaning that there can be only one RID for every distinct column value:

CREATE UNIQUE INDEX X ON T(A)

• A unique index can be created on a column that permits null values, Then there can be only a single row where the column is null.

I.e. the null value is treated like any other value. In contrast, primary keys and unique (alternative) keys can be created only on columns that are declared as NOT NULL. If the table already contains duplicates, the CREATE UNIQUE INDEX statement fails.

Indexes (3)

• Unique indexes are created internally for keys declared in the CREATE TABLE statement.

If one wants to select specific parameters for the index, one must first declare the table without the key, then create the necessary index, and then use ALTER TABLE to add the key. It checks whether there is already an index that can be used.

 One can add columns to the index that are not used in the search, e.g. that are not considered in the uniqueness condition:

CREATE UNIQUE INDEX X ON T(A) INCLUDE (B)

This permits more queries to be answered directly from the index, without fetching the rows via the RID.

4 - 8()

Indexes (4)

 The leaf nodes of a B⁺-tree index contain all column values in sorted order.

This can be used for merge joins, and for ORDER BY or GROUP BY.

• As in the ORDER BY statement, one can specify for each column, whether the smallest value is listed first (ASC) or last (DESC):

CREATE INDEX X ON T(A ASC, B DESC)

Normally, the leaf nodes are linked in both directions, so that reverse scans are possible (although this is the default, ALLOW REVERSE SCANS is often explicitly specified). If this is not needed, use DISALLOW REVERSE SCANS (probably saves one pointer, simplifies query optimization).

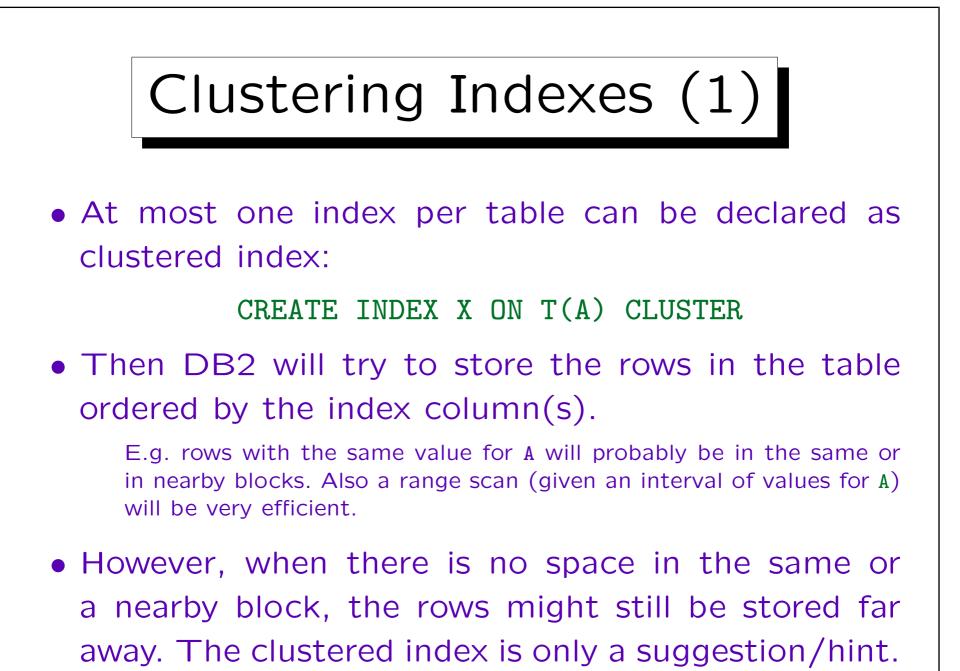
Indexes (5)

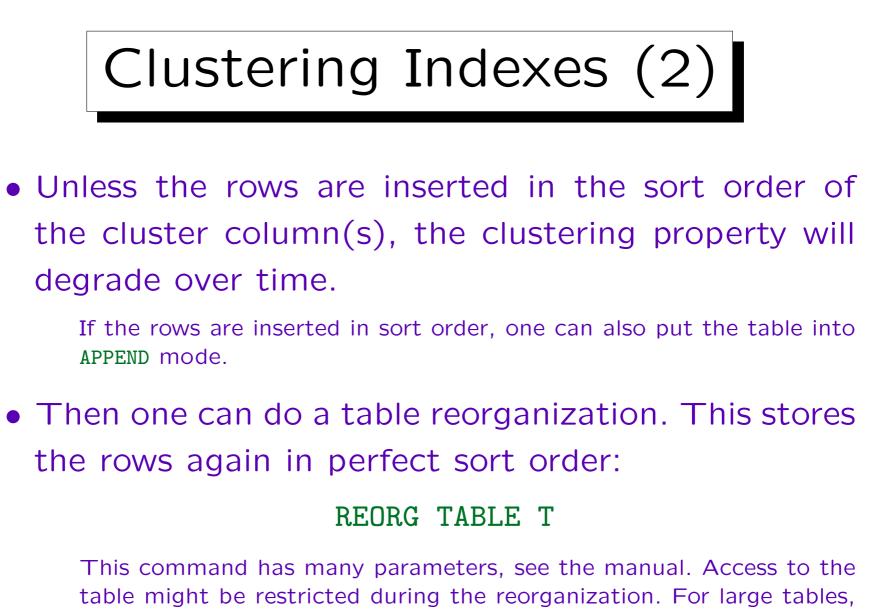
• Indexes are identified by schema and name (just like tables).

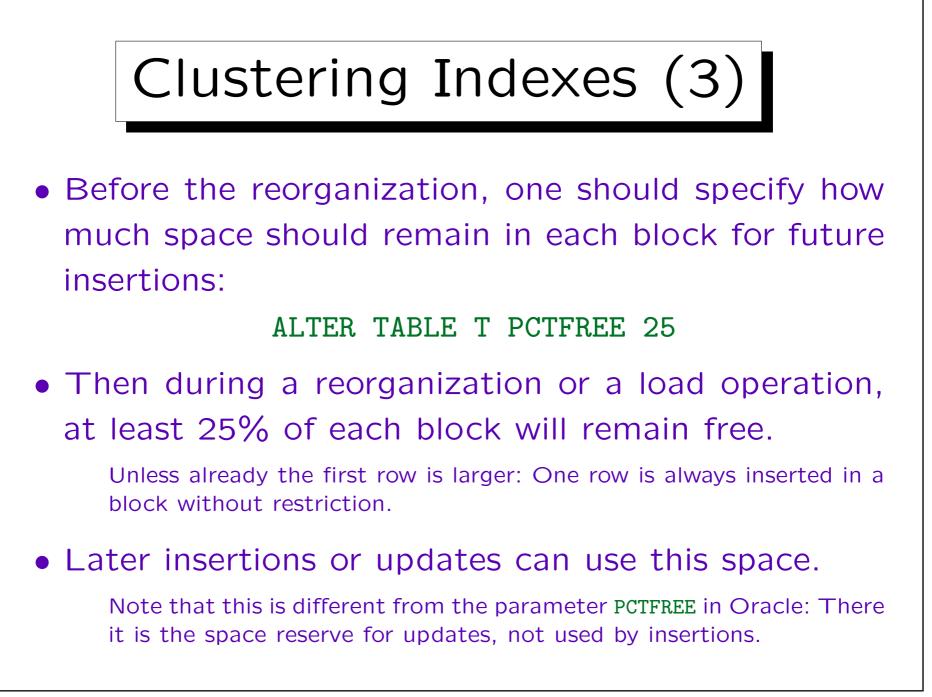
Different tables cannot have indexes with the same name (in the same schema). A table and its index can be in different schemas.

- Indexes and tables are in different namespaces: an index and a table with the same name are possible.
- Each index entry including all overhead can be at most 25% of the page size.

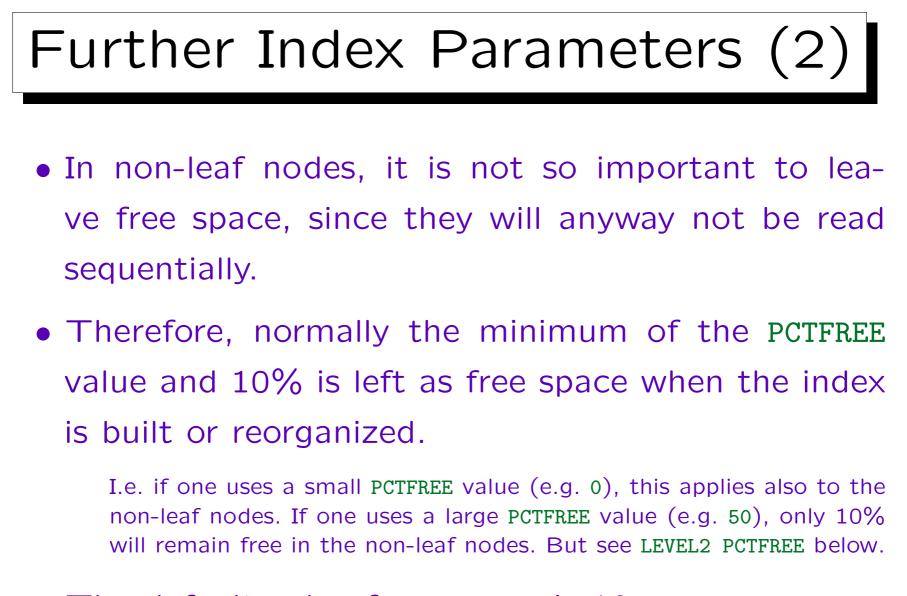
An index entry needs the same space as a row with the index columns would need plus typically 11 Bytes (for the RID).



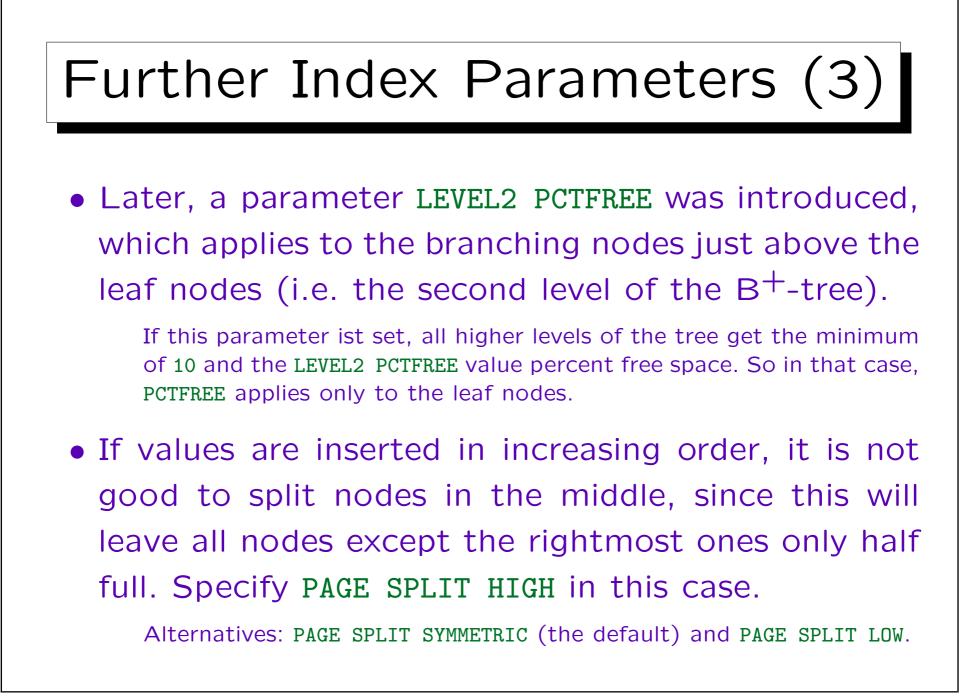


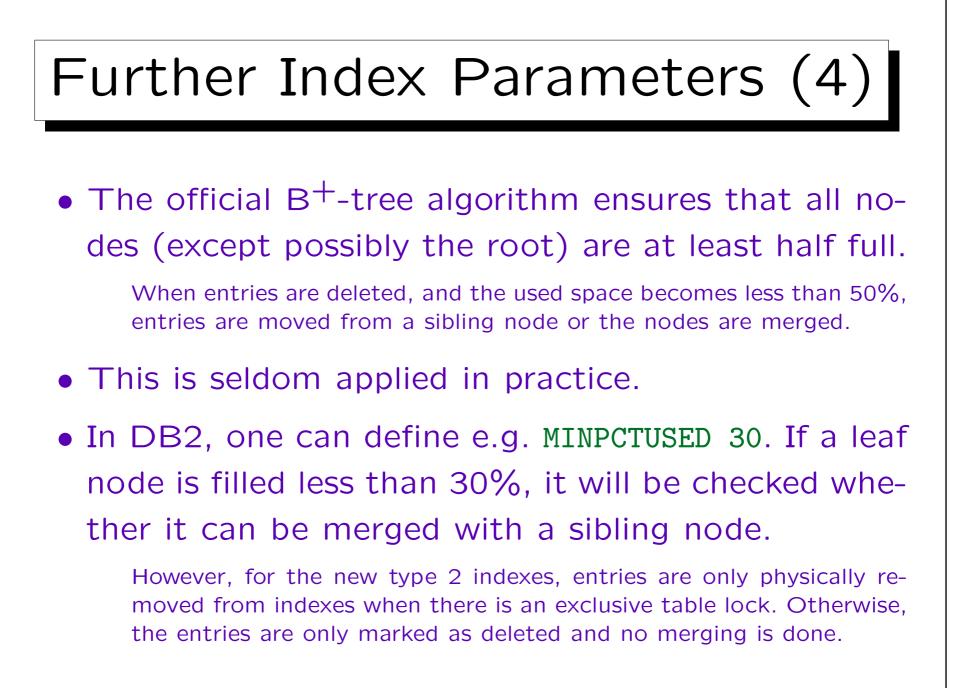


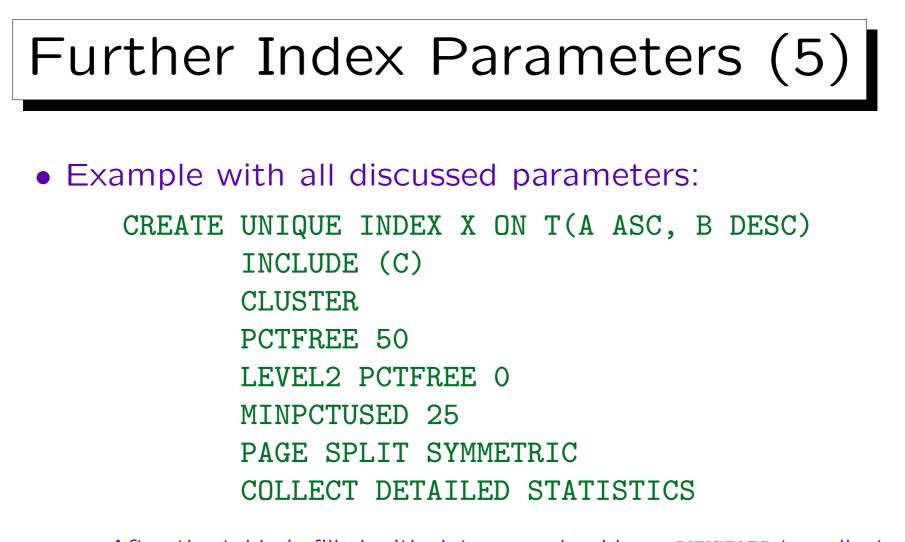
Further Index Parameters (1)
 Also indexes have a parameter PCTFREE: CREATE INDEX X ON T(A) PCTFREE 50
 When the index is created or reorganized, 50% of the space in the leaf blocks will remain free.
 Therefore, insertions are possible for some time wi- thout splitting the leaf nodes.
 In this way, the leaf nodes remain in sequential order on the disk (stored in contiguous blocks). This improves the performance of range scans or using the index for getting the column values in sorted order.



• The default value for PCTFREE is 10.



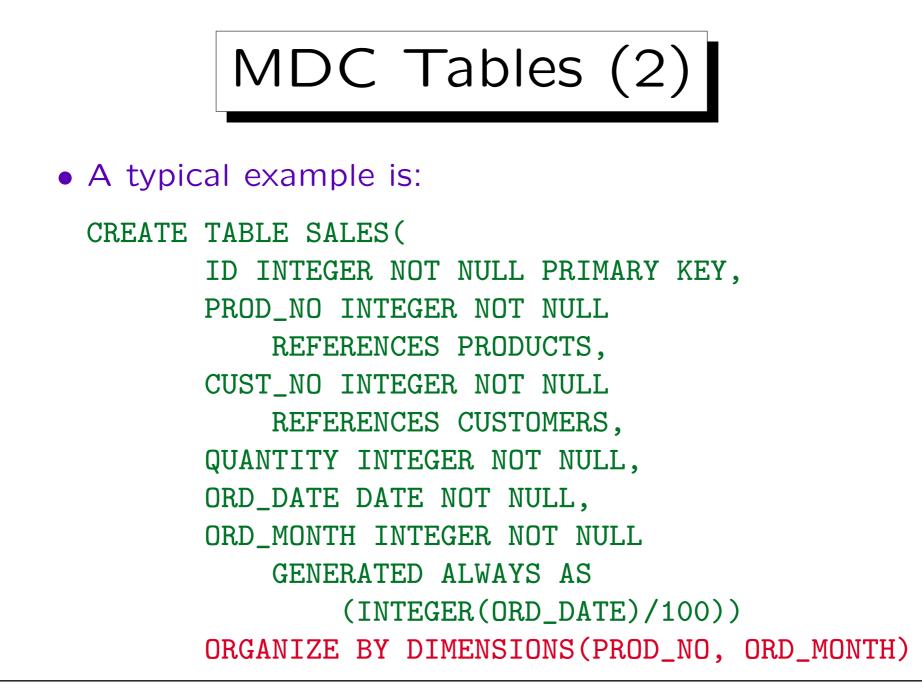


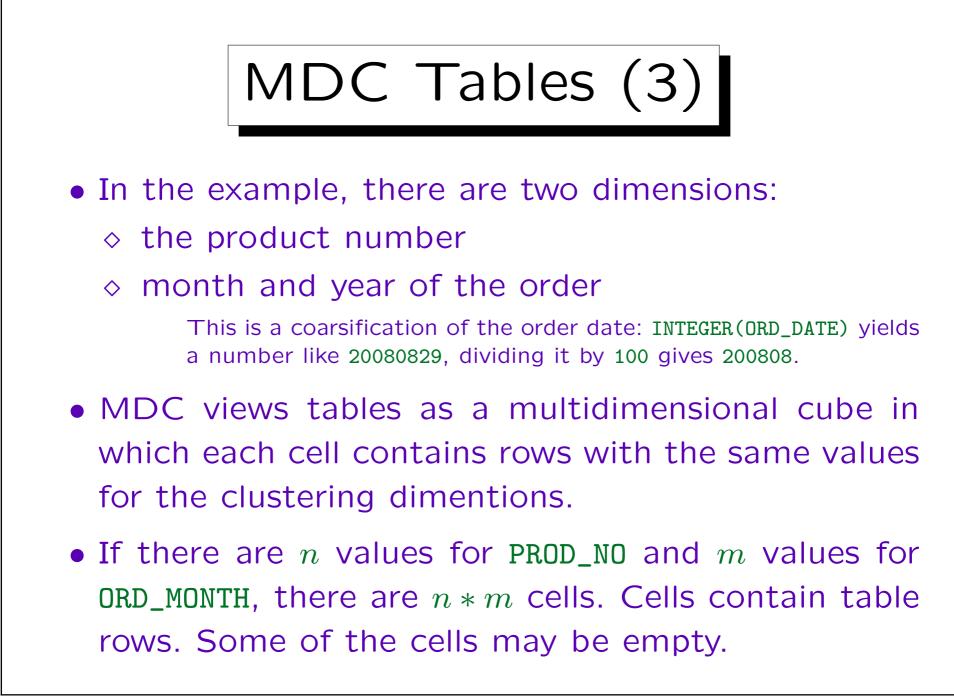


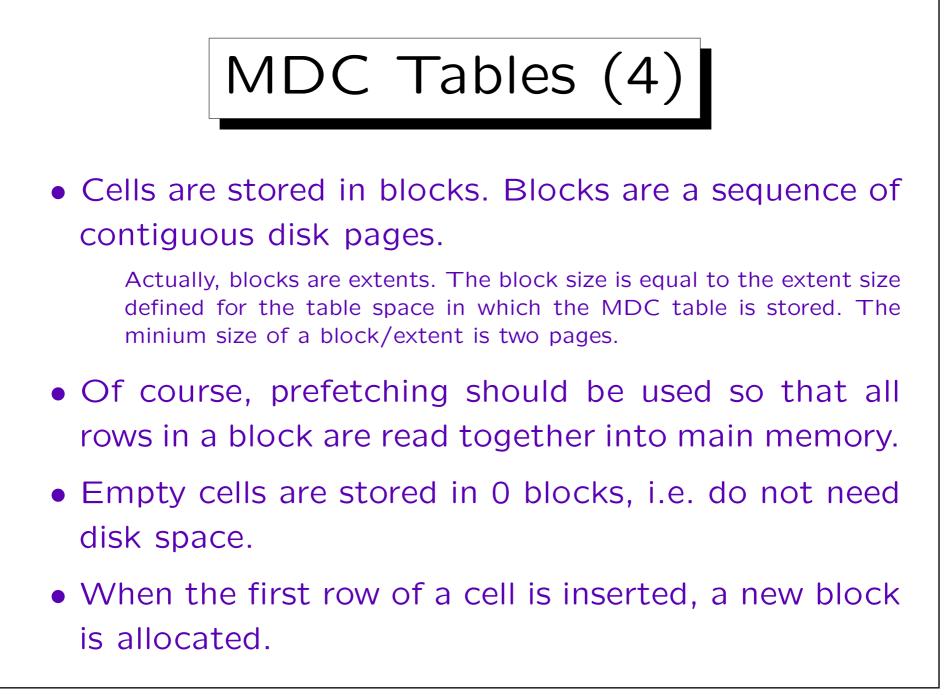
After the table is filled with data, one should use RUNSTATS to collect statistics for the optimizer. If the index is created when the table already contains data, one can use COLLECT STATISTICS instead.

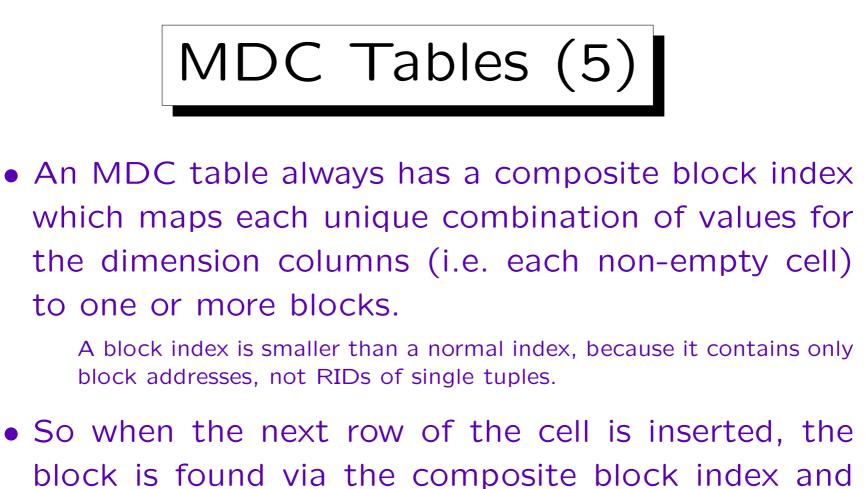


- Multidimensional clustering (MDC) is a new storage structure for tables (introduced in Ver. 8).
- It is primarily intended for data warehouse applications (OLAP), but it may also be useful for classical OLTP applications.
- Whereas the clustering implemented with cluster indexes needs periodic table reorganizations, the clustering in MDC tables is automatic.
- MDC tables apply block indexes which are smaller than standard indexes.

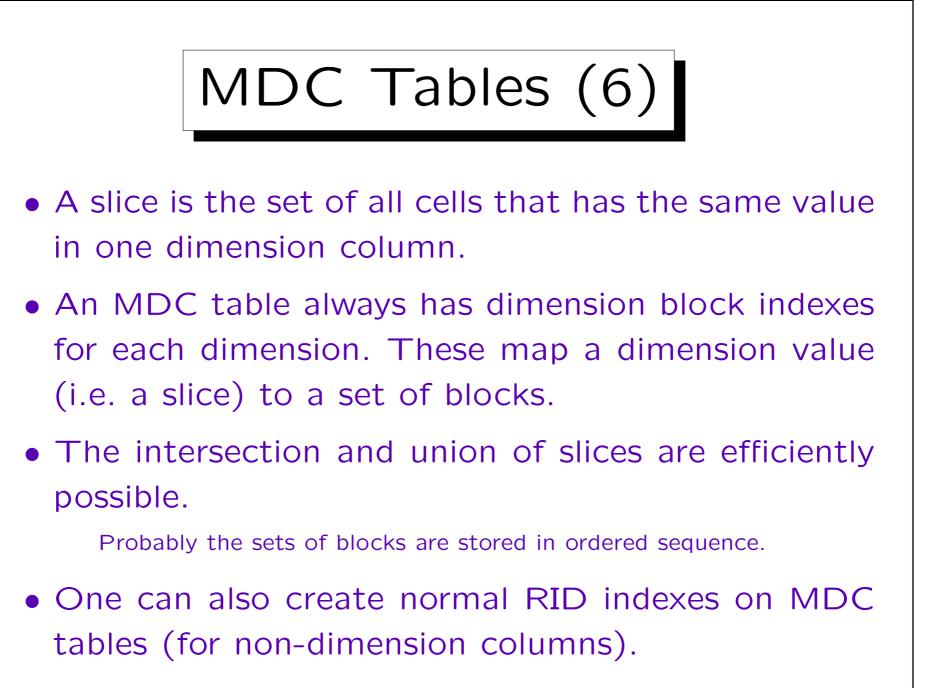


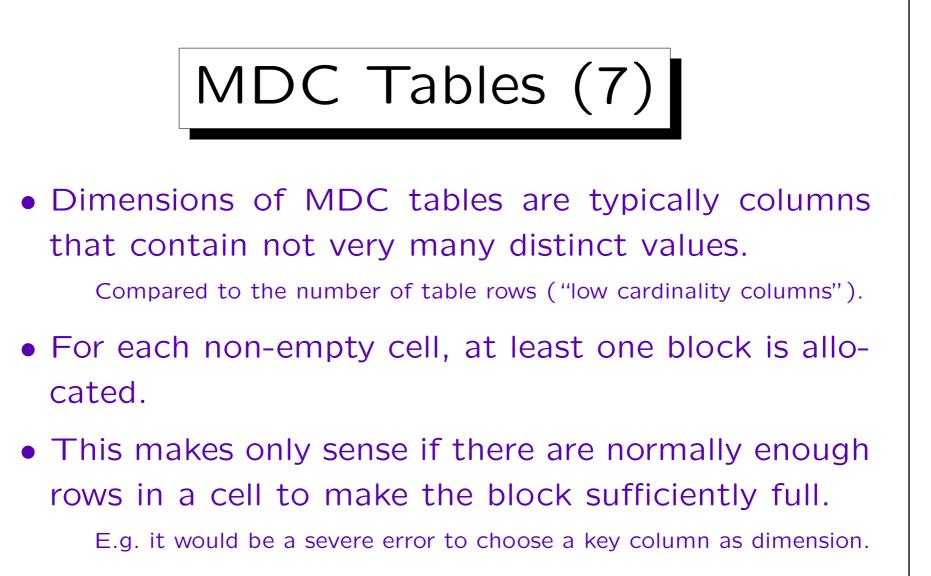






- the row is inserted into the same block.
- When the block is full, another block is allocated for the same cell (not necessarily nearby).





• As shown in the example, it might be possible to make the dimension granularity coarser ("rollup").