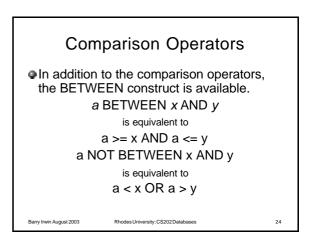
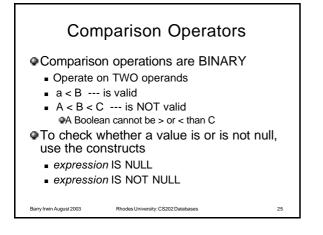


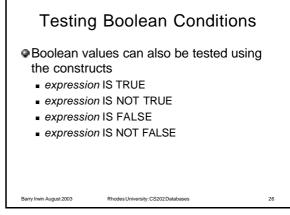
Logical Functions within SQL			
A	В	A AND B	A OR B
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
TRUE	NULL	NULL	TRUE
FALSE	NULL	FALSE	NULL
NULL	NULL	NULL	NULL
See PostgreSQL Function Ref Section 6.1 Barry Invin August 2003 Rhodes University: CS202 Databases 21			

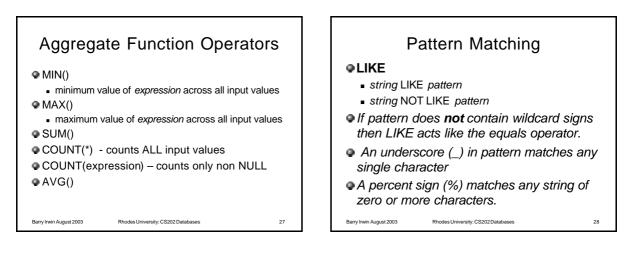
Log	ical Functio	ons within	SQL
	A	NOT A	
	TRUE	FALSE	
	FALSE	TRUE	
	NULL	NULL	
	L	1	J
See PostgreSQL Function Ref Section 6.1 Barry Irwin August 2003 Rhodes University: CS202 Databases 22			

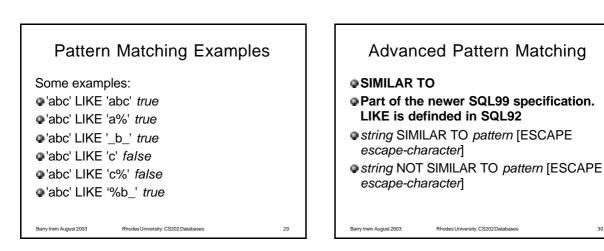
Comparison Operators		
Operator	Description	
<	Less Than	
>	Greater Than	
<=	Less Than or Equal	
>=	Greater Than or Equal	
=	Equal	
!= or <>	Not Equal	
Barry Irwin August 2003	See PostgreSQL Function Ref Section 6.2 Rhodes University: CS202 Databases 23	



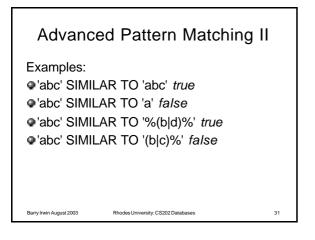




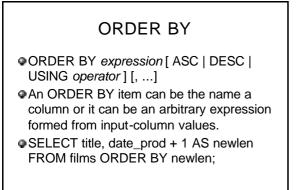




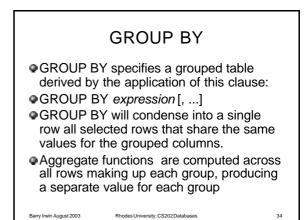




SC	QL arithmetic	
Examples		
	ntid, pracmark, testmark, estmark) AS classmark	
Barry Irwin August 2003	Rhodes University: CS202 Databases	32



Rhodes University: CS202 Databases



JMIT { count ALL } OFFSET start • where count specifies the maximum number of rows to return, and start specifies the number of rows to skip before starting to return rows. JMIT allows you to retrieve just a portion of the rows that are penerated by the rest of the query. f a limit count is given, no more than that many rows will be teurned. f an offset is given, that many rows will be skipped before starting to eturn rows. When using LIMIT, it is a good idea to use an ORDER BY clause hat constrains the result rows into a unique order. Otherwise you nill get an unpredictable subset of the query's rows • Eg, you may be asking for the tent through twentieth rows, but tenth through twentieth in what drearing? You don't know what ordering		LIMIT
where count specifies the maximum number of rows to return, and start specifies the number of rows to skip before starting to return rows. MIT allows you to retrieve just a portion of the rows that are retrieved by the rest of the query. I a limit count is given, no more than that many rows will be eturned. If an offset is given, that many rows will be skipped before starting to eturn rows. When using LIMIT, it is a good idea to use an ORDER BY clause that constrains the result rows into a unique order. Otherwise you will get an unpredictable subset of the query's rows E. g, you may be asking for the tenth through twentieth rows, but tenth through twenteth in with at ordering	Type N	
specifies the number of rows to skip before starting to return rows. b JMIT allows you to retrieve just a portion of the rows that are penerated by the rest of the query. c ia limit count is given, no more than that many rows will be eturned. c ia norfset is given, that many rows will be skipped before starting to eturn rows. c When using LIMIT, it is a good idea to use an ORDER BY clause hat constrains the result rows into a unique order. Otherwise you nill get an unpredictable subset of the query's rows n Eg. you may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering You don't know what ordering s	boolean	
ipenerated by the rest of the query. it is in the query. it is in the query. f a limit count is given, no more than that many rows will be eturned. it is given, no more than that many rows will be eturned. f an offset is given, that many rows will be skipped before starting to eturn rows. it is a good idea to use an ORDER BY clause hat constrains the result rows into a unique order. Otherwise you ill get an unpredictable subset of the query's rows it result rows, but tenth through twentieth in what ordering? You don't know what ordering	bytea	specifies the number of rows to skip before starting to return rows.
eturned. f an offset is given, that many rows will be skipped before starting to eturn rows. When using LIMIT, it is a good idea to use an ORDER BY clause hat constrains the result rows into a unique order. Otherwise you will get an unpredictable subset of the query's rows • Eg, you may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering? You don't know what ordering	characte	enerated by the rest of the query.
eturn rows. Vhen using LIMIT, it is a good idea to use an ORDER BY clause hat constrains the result rows into a unique order. Otherwise you vill get an unpredictable subset of the query's rows • Eg, you may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering? You don't know what ordering	characte	
when using Linking is a good need to use an OKDER of clause	,	
vill get an unpredictable subset of the query's rows • Eg, you may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering? You don't know what ordering	integer	Vhen using LIMIT, it is a good idea to use an ORDER BY clause
unloss you specify ORDER BY	numerio	hat constrăins the result rows into a unique order. Otherwise you vill get an unpredictable subset of the query's rows
unless you specify ORDER BY. te	smallin	 Eg. you may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering? You don't know what ordering
	text	unless you specify ORDER BY.
s	serial	

- € LI
- € LI ge o lf
- re If
- œ. re
- ø W th w

Barry Irwin August 2003

Barry Irwin August 2003

Rhodes University: CS202 Databases

35

33

SQL Data Types

Type Name	Aliases	Description	
boolean	Bool, logical	Boolean (T/F)	
bytea		binary data	
character	varying(n) varchar(n)	variable-length character string	
character(n)	char(n)	fixed-length char string	
date		calendar date (year, month, day)	
integer	int, int4	signed four-byte integer	
numeric [(p, s)]	decimal [(p,s)]	exact numeric with selectable precision	
smallint	int2	signed two-byte integer	
text		variable-length character string	
serial	serial4	autoincrementing four-byte integer	
Se Barry Irwin August 2003	e PostgreSQL Use Rhodes University: CS2		

Esoteric Data Types

 box cidr circle inet macaddr polygon 	rectangular box in 2D plane IP network address circle in 2D plane IP host address MAC address closed geometric path in 2D plane	
Barry Irwin August 2003		37

SQL Compatibility

 The following generic types are generally supported across all SQL implementations

bit, bit varying, boolean, char, character varying, character, varchar, date, double precision, integer, interval, numeric, decimal, real, smallint, time, timestamp

Rhodes University: CS202 Databases

38

Barry Irwin August 2003

Numeric Data Types

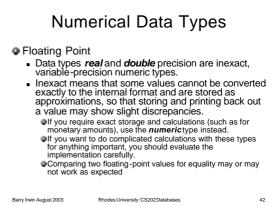
- store whole numbers, without fractional components
- The type *integer* is the usual choice
- best balance between range, storage size, and performance.
- The *smallint* type is generally only used if disk space is at a premium.
- The *bigint* type should only be used if the integer range is not sufficient

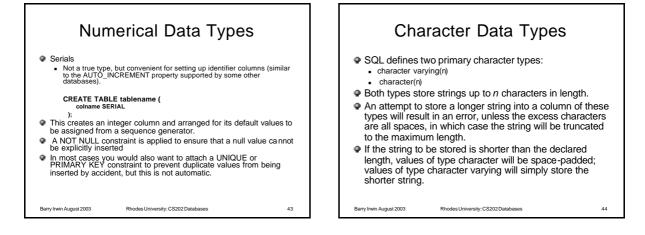
Barry Irwin August 2003

Rhodes University: CS202 Databases

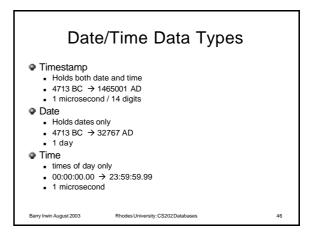
<section-header><section-header><section-header><list-item><list-item><list-item><list-item><list-item>

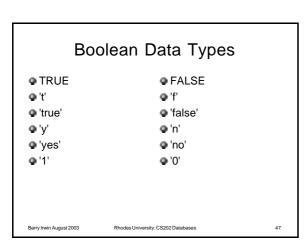
Numerical Data Types Numeric • The type *numeric* can store numbers with up to 1,000 digits It is especially recommended for storing monetary amounts and other quantities where exactness is required. • numeric type is very slow compared to the floatingpoint types Both the precision and the scale of the numeric type can be configured. NUMERIC(precision, scale) NUMERIC(precision) -- selects a scale of 0. Barry Irwin August 2003 Rhodes University: CS202 Databases 41

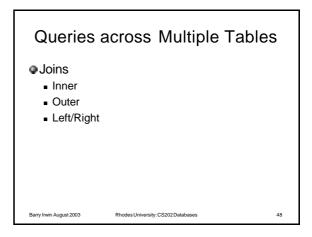


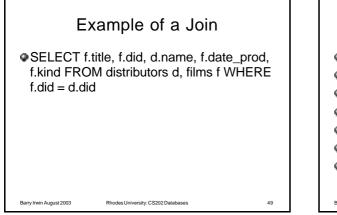


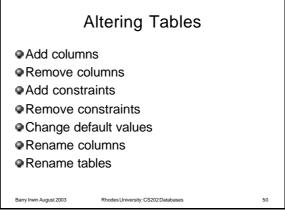
Date/Time Data Types			
January 8, 1999	unambiguous		
1999-01-08	ISO-8601 format		
1/8/1999 ; 8/1/1999	U.S. vs European mode		
1999.008 ; 99008	year and day of year		
19990108	ISO-8601 year, month, day		
990108	ISO-8601 year, month, day		
J2451187	Julian day		
January 8, 99 BC	year 99 before the Common Era		

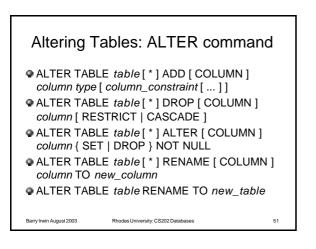


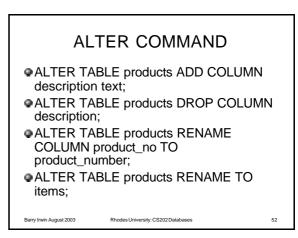




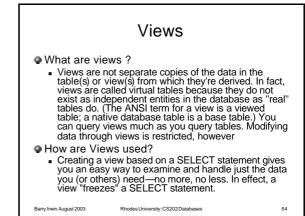




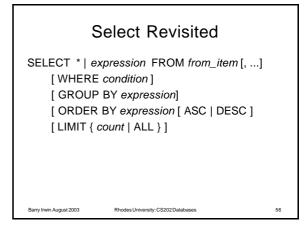


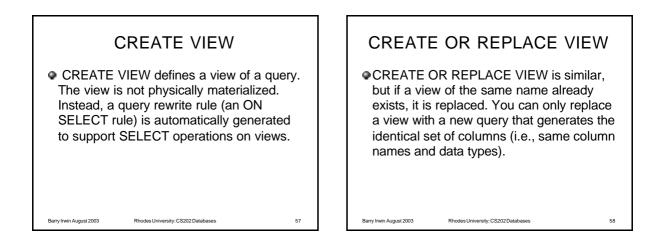


Cr	eating Tables (2)	
	NTO create a new table from sof a query	om
INTO TAB [FROM from_	ession [AS output_name] [,] LE new_table _item [,]] [WHERE condition] [′ expression	
Barry Irwin August 2003	Rhodes University: CS202 Databases	53

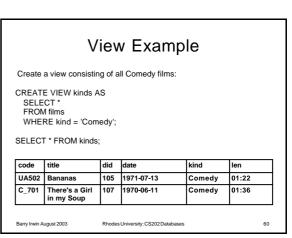


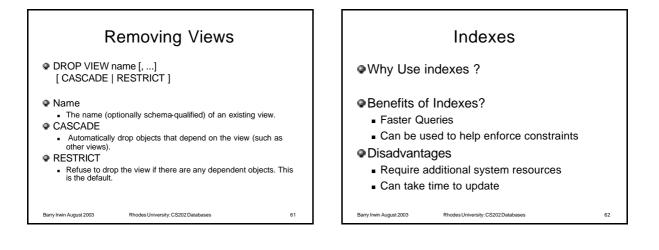
	Creating views	
	REPLACE] VIEW view name list)] AS SELECT query	
View	tionally ophone suplified) of a view to b	a areated
 The name (o column name 	ptionally schema-qualified) of a view to be list	e created.
 An optional list given, these is 	st of names to be used for columns of the names override the column names that we the SQL guery.	
Query		
	y (that is, a SELECT statement) which wi and rows of the view.	ll provide
Refer to SELEC	CT for more information about valid argun	nents.
Barry Irwin August 2003	Rhodes University: CS202 Databases	55

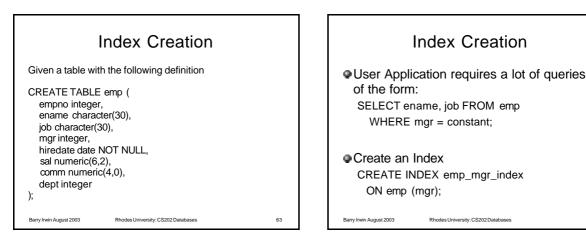


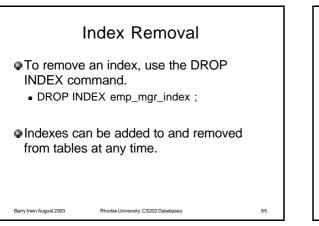


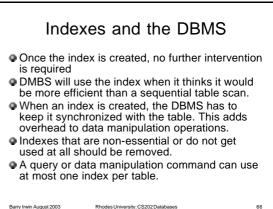
М	ore on Views	
not allow an in You can get th creating rules t view into appro	ws are read only: the system will sert, update, or delete on a view le effect of an updatable view by that rewrite inserts, etc. on the opriate actions on other tables. mation see CREATE RULE.	
Use the DROF	VIEW statement to drop views.	
Barry Irwin August 2003	Rhodes University: CS202 Databases	59

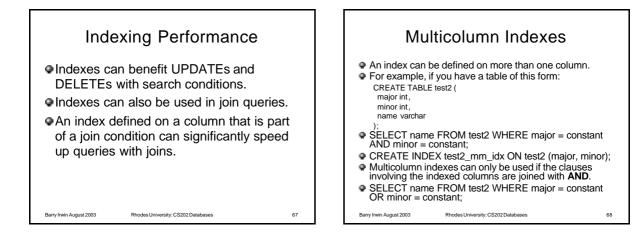


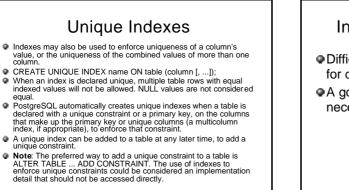












Barry Irwin August 2003

Barry Irwin August 2003

Rhodes University: CS202 Databases

Index Performance Testing
Difficult to formulate a general procedure for determining which indexes to set up.
A good deal of experimentation will be necessary in most cases.

Rhodes University: CS202 Databases

Barry Irwin August 2003

69

71

Index Performance Testing up indexes will tell you what indexes you need for the test data, but that is all.
It is especially fatal to use proportionally reduced data sets. While selecting 1000 out of 100000 rows could be a candidate for an index, selecting 1 out of 100 rows will hardly be, because the 100 rows will probably fit within a single disk page, and there is no plan that can beat sequentially fetching 1 disk page.
Be careful when making up test data, which is often unavoidable when the application is not in production use yet. Values that are very similar, completely random, or inserted in sorted order will skew the statistics away from the distribution that real data would have.

Rhodes University: CS202 Databases

Advanced Indexes

- PostgreSQL provides several index types:
 - B-tree
 - R-tree
 - GiST
 Hash.
- Each index type is more appropriate for a particular query type because of the algorithm it uses.
- By default, the CREATE INDEX command will create a B-tree index
- The PostgreSQL query optimizer will consider using a B-tree index whenever an indexed column is involved in a comparison using one of these operators:
- R-tree indexes are especially suited for spatial data. The PostgreSQL query optimizer will consider using an R-tree index whenever an indexed column is involved in a comparison using one of these operators:
- <<, &<, &>, >>, @, -=, &&
 The query optimizer will consider using a hash index whenever an indexed column is involved in a comparison using the = operator.

Barry Irwin August 2003

